IDA

INSTITUTE FOR DEFENSE ANALYSES

Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms

Edward A. Schneider, Task Leader

Edward A. Feustel
Ronald S. Ross

November 1997

IDA Paper P-3375
Log: H 97-003597

This work was conducted under contract DASW01 94 C 0054, Task T-AA5-1409, for the National Security Agency and the Defense Information Systems Agency. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

INSTITUTE FOR DEFENSE ANALYSES

Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms

Edward A. Schneider, Task Leader

Edward A. Feustel
Ronald S. Ross

Preface

This document was prepared by the Institute for Defense Analyses (IDA) under the task order, Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms. It fulfills this task objective to provide a final report containing the results of our investigation. The work was sponsored by the National Security Agency (NSA) and Defense Information Systems Agency (DISA).

The following IDA research staff members were reviewers of this document: Mr. John Boone, Dr. Alfred Brenner, Dr. Richard Ivanetich, Mr. Terry Mayfield, Dr. Reginald Meeson, Dr. Richard Morton, and Mr. Stephen Welke. The document also benefited from the comments of Ms. Sylvia Reynolds.

The contributions of Dr. David Gomberg, formerly of the MITRE Corporation, Mr. Bill Smith of DISA, and Mr. Rob Rosenthal of the Defense Advanced Research Projects Agency (DARPA) are gratefully acknowledged.

Table of Contents

List of Figures

List of Tables

Executive Summary

This report presents the results of a study of the Department of Defense (DoD) Goal Security Architecture (DGSA). In March 1995, the Assistant Secretary of Defense for Command, Control, Communication, and Intelligence (ASD-C3I) established the DGSA, as part of the TAFIM, as a framework for security architectures within DoD. However, the lack of systems that support this framework has hindered its acceptance. Indeed, many security engineers believe that an implementation is impossible on currently available computing systems. This study by the Institute for Defense Analyses, under tasking from the National Security Agency and the Defense Information Systems Agency, investigated the accuracy of this belief. A side effect of this study was the discovery of several areas in the DGSA report where the concepts are not clear; this report includes our interpretation. The study used Trusted Information Systems' Trusted Mach (TMach) operating system as the basis for a test implementation: TMach appears to provide the best support, of the commercially available trusted systems, for the DGSA.

The results in this report include:

- Formal definitions for concepts defined in the DGSA and a model that describes distributed system implementations.
- Extensions of DGSA concepts for features needed in the computational process.
- A demonstration of how to create a DGSA-style architecture by first describing a mission statement and then developing a security policy and a set of components that support the mission.
- The services needed by security management.
- A description of TMach and the allocation of TMach mechanisms to the services required by the security policy.
- Areas of the DGSA that current operating systems like TMach do not support, areas of the DGSA that are hard to implement, and actions that can be taken to support the use of the DGSA to build secure systems.

The primary conclusion of the study is that a system like TMach, based on the client-server paradigm, can support most of the DGSA. In particular, the design properly separates different security policy domains and provides some separation between security policy definition and implementation. Areas where this design is lacking include:

- Lack of generality of the separation between policy definition and implementation.
- Lack of an evaluation methodology that leads to assurance of absolute protection.
- Lack of the security association that distributes a security domain over several hardware platforms.
- Lack of an operating system personality that supports popular commercial, off-the-shelf (COTS) software.
- Lack of trusted servers that support intra-domain and inter-domain security policy.

The effort to correct these areas is outside the scope of this study. Also, these weaknesses relate only to support for the DGSA, not to TMach's suitability as a trusted operating system.

A second general conclusion of the report is that the DGSA offers an appropriate conceptual framework for the expression of security policies and for the design of systems of systems that implement those policies. We encountered no difficulty expressing a guard example policy in terms of DGSA concepts, nor is there difficulty expressing a policy equivalent to Multi-level Security (MLS).

A third general conclusion is that the DGSA provides a workable basis for the construction of implementable systems. An exception to this conclusion occurs when the security policies for the information domains differ greatly from those currently used or contemplated. Thus, a full implementation requires substantial additional work, as the report demonstrates. Since much of the additional work is in support of arbitrarily complex security policies, standardizing on a few parameterizable policies might cover the majority of useful cases and present less difficulty in operational accreditation and operational assurance. Further, the activity responsible for DGSA should develop guidance that aids the implementer but does not overly constrain his implementation.

1. Introduction

1.1 Background

Information technology has evolved from a collection of mainly independent information systems to an internetworking of systems in which information is easily shared, but which also provides avenues for abuse from outside an organization and even outside national boundaries. The ability of the Department of Defense (DoD) to accomplish its missions has become dependent on this technology to facilitate the sharing and exchange of information among DoD elements and also with coalition partners. In such an environment, protecting the integrity, availability, and confidentiality of information is crucial to the successful accomplishment of DoD's mission.

In 1993, a new set of requirements[1] was identified which would lead to a major alteration in DoD's approach to information security and its policies governing information systems. These requirements state that DoD information systems must:

1. Support information processing under multiple security polices of any complexity or type, including those for sensitive unclassified information and multiple categories of classified information.
2. Be sufficiently protected to allow distributed information processing (including distributed information system management) among multiple hosts on multiple networks in accordance with open system architectures.
3. Support information processing among users with different security attributes employing resources with varying degrees of security protection, including users of non-secure resources if a particular mission so dictates.
4. Be sufficiently protected to allow connectivity via common carrier (public) communications systems.

---

1. *Department of Defense (DoD) Information Systems Security Policy, DISSP-SP.1* [4].

Further supporting these requirements, the Chairman of the Joint Chiefs of Staff recently issued the following policy[2] with respect to Defensive Information Operations:

a. Information, information-based processes, and information systems (such as command, control, communications, and computer (C4) systems, weapon systems, and infrastructure systems) used by US military forces will be protected relative to the value of the information contained therein and the risks associated with the compromise of or loss of access to the information.

b. Information system defense relies on four interrelated processes. These include a process to protect information and information systems, a process to detect attacks or intrusions, a restoration process to mitigate the effects of incidents and restore services, and a response process.

The DoD has developed an evolutionary goal security architecture that it believes satisfies the requirements and policy outlined above: The DoD Goal Security Architecture (DGSA)[3]. The DGSA defines terms relating to those concepts and describes several fundamental security concepts that must exist in future security architectures in order to meet the requirements outlined above.

The *information domain*, the centerpiece of the security concepts, is a means for specifying how information must be managed and controlled within information systems. It organizes information that is to be shared by a group of users that have agreed to a security policy, enforced by the system, on the use of that information. An information domain provides logical boundaries for uniform protection policies. The information domain concept addresses the management and control of information from a logical perspective, independent of the physical information systems and environments where processing, transmission, and storage occur.

Information domains support specific missions that require use of the information in them. Specific organizations use and/or control information systems supporting the information domains. Thus, conceptually, the separation of an information domain from any particular systems processing environment is a key aspect of the DGSA approach to information security. Understanding the structural characteristics and behavior of domains and systems with respect

---

2. Office of the Chairman of the Joint Chiefs of Staff. *Defensive Information Operations Implementation*. Document CJCSI 6510.01B, August 1997.

3. The DGSA was developed by the Defense Information Systems Agency's Center for Information System Security (now called the INFOSEC Program Management Office, IPMO) as part of the Defense-wide Information System Security Program. It is a key part of a larger DoD enterprise architecture described in the Technical Architecture Framework for Information Management (TAFIM) [6].

to information security provides the basis for linking the logical and physical components of information management in an operational environment.

1.2 Purpose

To date, no implementation of a system embodying all aspects of the DGSA has been delivered, even though the DGSA received official status when the Technical Architecture Framework for Information Management (TAFIM) was approved over two years ago[4]. In 1996 the National Security Agency (NSA) and the Defense Information Systems Agency (DISA) collaborated in tasking the Institute for Defense Analyses (IDA) to investigate whether a commercially available operating system could serve as the base platform for DGSA. If not, IDA was to determine what additional functionality might be required of the operating system, or what changes might be required or recommended for DGSA.

The DGSA document presents a very abstract view of a security architecture. This view provides a set of constraints that can be used to eliminate non-conforming implementations from the set of implementations that are permissible. Consider the tree formed by making implementation decisions. Each branch of the tree is the result of a design decision. Each leaf on the tree represents a possible implementation. The purpose of DGSA is to cut off all leaves that are non-conforming by trimming branches that are non-conforming. Ordering decisions properly will guarantee a tree with the largest number of conforming implementations. Thus DGSA affects both the logical architecture and the systems architecture. The paper addresses both architectures.

In addition, the paper suggests a possible integration of technologies that would lead to an implementation of the DGSA architecture in a practical form. One of the technologies is Trusted Information Systems' (TIS) Trusted Mach (TMach®) operating system. At the time the task was initiated, TMach was to have been evaluated at the B3 level of trust in the United States and at the B3-E5 level of trust[5] in Europe. Further, it was to have been sold commercially and could have provided a basis for Unix and for DOS applications. Finally, of the systems for which adequate documentation is available, TMach appears best suited to support the DGSA. The paper provides a description of TMach and shows how this system may be modified to support a mission.

---

4. Emmett Paige, Jr. *Technical Architecture Framework for Information Management (TAFIM), Version 2.0.* ASD-C3I Memo, March 1995.

5. The B3 level of trust refers to a specific standard defined in the U.S. Trusted Computer System Evaluation Criteria (TCSEC).The B3-F5 level of trust refers to a specific standard defined in the European Information Technology Evaluation Criteria (ITSEC).

A second required technology provides Security Associations (SAs), allowing information to be shared between systems controlled by different organizations. Currently the Internet Engineering Task Force (IETF) is engaged in prototyping and specifying the Internet Protocol (IP) Security (IPSEC) RFC for Internet Protocol Version 6, which incorporates a successor to the Internet Security Association and Key Management Protocol (ISAKMP) developed by NSA. Implementations of IPSEC are available for test purposes on the LINUX operating system and on CISCO Systems routers. These implementations of the IPSEC specification, and others in the future, will provide many of the pieces required to implement security associations between instantiations of an information domain on multiple platforms. Combinations of the two technologies could provide a basis for a complete DGSA implementation.

1.3 Audience

This paper is written for those who need to understand the DGSA and who need to apply it to the implementation of information systems. The authors assume that the reader has an understanding of the basic principles of operating systems, digital communications, and computer security, and is familiar with the DGSA Version 3.0 document.

1.4 Organization of the Paper

In order to make use of this "architecture", the implementor needs a very precise understanding of the terms used in the DGSA. The authors, with the help of many reviewers, have developed a set of formal definitions that we believe capture the intent of the DGSA and that help to set those ideas in a context of implementation. The implementation-independent formal definitions are presented in Chapter 2.

Chapter 3 provides the means for linking the definitions of Chapter 2 to a networked environment. The DGSA definition [5] omits many details required to determine whether an implementation is sufficient or even feasible. We therefore formalize and extend the DGSA by providing an abstract model for information management within the framework of the DGSA on actual computing systems. The abstract model interprets the high-level concept of information domains as implemented within a distributed systems environment. The model is intended to (1) facilitate understanding of the interactions between information domains and information systems, (2) provide a vehicle for elaborating the specific security concepts of the DGSA, and (3) permit the identification of issues associated with the implementation of the DGSA concepts.

Chapter 4 provides an example of the use of the DGSA in support of a mission to be performed by an information processing system. An application-level security policy statement

is viewed from the standpoint of the threats to the mission and the services required in order to perform the mission. A set of system requirements is derived from the example policy and a set of information domains required to implement the policy are developed. A security management information base supporting this set of domains is described.

Chapter 5 describes the security services used in the management of a DGSA-style architecture, focusing on details required for the implementation of the mission described in Chapter 4.

Chapter 6 describes the organization of TMach and its use of mechanisms inherited from the Mach operating system. Components of TMach relevant to security are discussed.

Chapter 7 describes how the DGSA can be implemented using the TMach mechanisms described in Chapter 6 to support the security policy of Chapter 4 and to implement security features discussed in Chapter 4 and Chapter 5.

Chapter 8 provides our findings. First it discusses TMach and its usability. Next it summarizes our experience on the use of TMach to realize a DGSA compliant architecture. Finally, it gives our recommendations on the further development of the DGSA based on our experience obtained in fitting TMach to DGSA requirements.

Several appendices are included to provide the reader with references and additional bibliography which may be used for further study of DGSA and security as seen from the International Standards Organization perspective.

2. System Independent DGSA Concepts

Information management can be characterized by several interrelated components: security policy, controlled entities, information domains, and information systems. These components play a significant part in defining effective protection schemes for controlling and managing information and information system resources. This section defines and relates each of the components, and addresses the relationship between information domains and the information systems.

The DGSA deals with information objects as artifacts of computation. Because the DGSA does not deal with a model of security that must be enforced during computation and that governs the creation, use, and destruction of transient objects, this document augments the DGSA with a collection of invented computational components[6]. The computational model is consistent with the defining document at those points in time when a computation has completed and does not violate the security policy in force. An argument can be given that the security policy is not violated during the computation: this is beyond the scope of this document.

2.1 Security Policy

Given that information and information system resources have some associated value and are worth sharing and protecting, specific security requirements can be defined by a process that (1) identifies the information and/or information system resources to be managed, (2) states the operational requirements for their use, (3) categorizes their value, (4) identifies potential threats to them, and (5) specifies the measures that must be taken to provide adequate protection from potential threats. The security requirements are derived from high-level security objectives and articulated through security policies. The security policies discussed are policies derived from high-level (application-level) requirements[7].

> **Definition 2.1:** A *security policy* $p \in \mathbf{P}$, where $\mathbf{P}$ represents the universal set of all security policies, is a collection of rules designed to enforce an

6. These invented components are marked with a ✳

7. In Chapter 4 we will use this process to derive implementation-level requirements.

organization's desired protection for its information and other system resources, and to counter given threats.

Security policies provide direction, define responsibilities, and establish accountability. They can be developed at different levels of abstraction, ranging from high-level national policy to specific enterprise policies supporting missions and organizations. If the enterprise mission and organizational structure contain hierarchical relationships, demonstrating that any security policy within the hierarchy faithfully supports any higher-level policy should be possible.

Ultimately, when some form of computation or information processing is required, security policies must be refined into requirements on information system components for specific security services and protection mechanisms using a combination of hardware, software, and firmware solutions.[8] The protection mechanisms, in turn, exercise a degree of control over permissible operations (e.g., create, initialize, destroy, copy, regrade, export, import, sign, unsign, catalog, uncatalog, grant, revoke, share, unshare, update, and invoke) on information and information system resources, based on the requirements derived from the security policies. Security policies must also address protection requirements that provide the environmental boundary of the information system. These requirements involve physical, personnel, and procedural security components that can be selectively combined and controlled.

Specification of security policies necessitates the identification and designation of specific entities for which some degree of protection must be provided. Protection, in this situation, implies applying protection mechanisms to information or information system resources based on some pre-established criteria. Such criteria can be used statically or dynamically in an information system.

2.2 Entities and Attributes

The real world can be modeled by defining a collection of primitive objects called *entities* and specifying relationships among these objects. An entity is an object that exists and that can be distinguished from other objects. In general, there are three distinct types of entities that play a significant role in information management. These entities form the primitive elements from which all other structures are built. The first type of entity is information in its structural or object form.

---

8. The DGSA definition of security services is based on ISO 7498-2[9], ISO7498-4[10], ISO10181 [11-17] and includes authentication, access control, data integrity, data confidentiality and non-repudiation. The DGSA definition also includes the security service of availability. Protection mechanisms in hardware, firmware, and software are employed to implement specific security services. Management services are also discussed.

Definition 2.2: An *information object* $o \in \mathbf{O}$, where $\mathbf{O}$ represents the universal set of all information objects, is a unit of information that is atomic with respect to security policies. The type of an information object defines the effect of the various accesses allowed by the security policy.

Information objects are used by processes conducting mission-oriented activities within an information system on behalf of a user.[9] Therefore, the second type of entity is the user. A user is represented within the information system by an information object containing the credentials of the user, which may include the user's identity, an authentication symbol for that identity, permissible groups and roles, and a set of rights for that user.

Definition 2.3: A *user* $u \in \mathbf{U}$, where $\mathbf{U}$ represents the universal set of all users, is a unit of responsibility for accesses to information objects and information system resources. Each user is represented by an information object containing its credentials.

Information is assumed to be created, accessed, updated, and disseminated within a framework of shared information system resources to include supporting hardware, firmware, and system software. Thus, the third type of entity is the information system resource[10].

Definition 2.4: An *information system resource*$* r \in \mathbf{R_t}$, where $\mathbf{R_t}$ represents the set of information system resources at time t, is a provider of information system services and/or facilities for processing, transmission, and storage (e.g., input/output devices, memory, registers, system functions).

Entities are distinguished from one another by associating a set of *attributes* with each that describes its properties or characteristics. Attributes associated with an entity may be subsequently associated with a value, thus completing the description. For example, a user may have a set of attributes consisting of name, social security number, and date of birth, with a set of assigned values. An information system resource, such as a computer workstation, may have a different set of attributes consisting of serial number, location, and memory capacity, instantiated with appropriate values.

Certain attributes and their assigned values can provide this distinction and serve as unique identifiers analogous to primary keys in a relational database. Other entity attributes

9. The DGSA concept of *user* is similar to that of *principal* elsewhere [1] and may be an identifiable individual, a role that any of several individuals may perform, a system function created when the system is initialized, or one of these users operating with restricted permissions. The representation of individuals within an information system occurs through a controlled login procedure that involves identification and authentication of the prospective user. Upon successful completion of the login procedure, processes (programs in execution) are created to act on behalf of that user within the system.

10. An *information system resource* may be multiplexed between different security contexts, but at any given time is included in the pool of resources owned by the system or is in exactly one security context.

from the general set of attributes may be useful in expressing functional characteristics. Still other entity attributes from the general set of attributes may be useful in expressing the relevant characteristics necessary for supporting information security policies. These security-related attributes are described below.

2.2.1 Security Attributes

Selected attributes are termed security attributes if they are relevant to the application of particular security policies (e.g., confidentiality, integrity, need-to-know).[11] An information system security policy decision function uses these attributes to determine whether certain requested uses of information and information system resources (e.g., reading from a file, writing to a disk, executing a program) should be permitted by the security policy enforcement function.

> **Definition 2.5:** A *security attribute* $a \in \mathbf{A}$, where $\mathbf{A}$ represents the universal set of all security attributes, is an element of information that is associated with an entity for the purpose of applying a security policy. Each security attribute has an associated set of values.

There are many types of security attributes. For example, a user may have a security attribute related to secrecy indicating the type of information the individual is authorized to access. An information object, such as a file, may have a security attribute related to integrity indicating the quality or assurance associated with the process that created it. Also, an information system resource, such as a laser printer, may have a security attribute indicating the security level of information authorized to be printed on that device.

Recall that entities include users, objects, and system resources. The definition of entities can now be extended to include security attributes and be made more formal.

---

11. There is an implication in this list that confidentiality, integrity, and need-to-know are separate policies. While speaking of such policies (as do the ISO Security Frameworks [13]) can be useful when applied to (a portion of) an enterprise, one security policy may specify a number of security requirements. That is, a security policy may have protection requirements involving non-disclosure, integrity and need-to-know or any number of other requirements (e.g., non-repudiation). The space of applicable requirements is substantially larger than that traditionally associated with security and is necessitated by Requirements 1 and 3.

Definition 2.6: Let A_e be the set of possible values of the security attributes for entity *e* and

$$E_{OBJECT} = \{ (o, a) \mid o \in \mathbf{O} \wedge a \in A_o \},$$

$$E_{USER} = \{ (u, a) \mid u \in \mathbf{U} \wedge a \in A_u \},$$

$$E_{RESOURCE}(t) = \{ (r, a) \mid r \in \mathbf{R_t} \wedge a \in A_r \}.$$

Then a *controlled entity** $e \in (E_{OBJECT} \cup E_{USER} \cup E_{RESOURCE}(t))$ is an information object, user, or resource that has associated security attributes.

2.2.2 Entity-Attribute Constraint Relationships

Given the definitions of controlled entities and security attributes, relationships between arbitrary groups of entities can be formally defined based on a set of well-defined constraints established by a security policy. Let *f* represent a proposed action to carry out a selected applicable operation with respect to a set of controlled entities. Then, a general decision function can be defined as:

$$d_f(E_U, E_O, E_R, t) \rightarrow decision(t)$$

where $E_U \subseteq E_{USER}$, $E_O \subseteq E_{OBJECT}$, and $E_R \subseteq E_{RESOURCE}(t)$[12].

The result of the decision function is a boolean value representing either grant-permission or deny-permission. The decision is based on (1) the protection requirements established by the security policy, (2) the set of decision rules generated from the security policy, (3) the value of the controlled entities' associated security attributes used by the rules, (4) accumulated information from previous security policy decisions[13], and (5) the environment of the information system, e.g., the time at which permission is requested, the operation is to be performed, or both. Thus, the decision can be interpreted as a determination of whether a specific user is permitted to use a designated information object(s) using specific information system resource(s), at a given time, under particular circumstances, and when operating under a spe-

12. Any of the controlled entity sets in the decision function argument (i.e., E_U, E_O, E_R) could be empty depending on the type of action represented by the function, *f*, and the protection requirements established by the security policy. One of the entities from E_O and E_R may contain state information conditioning *f*, e.g., that one of the objects is signed by a given user or by a list of users. This information corresponds to case 4 below.

13. This context information may be stored with the objects or with other metainformation about the state of the information domain.

cific security policy. The decision may cause the creation or modification of state information, represented as information objects or as information resources, that will be used in future decisions.

For example, an organization employing a confidentiality security policy may wish to control employee access to selected company files as well as the particular information system resources that may be used in the processing, transmission, or storage of those files. Therefore, when the employee wishes to print a sensitive file, the decision function d could use a controlled-entity triple as its argument that includes the user, designated file, and authorized printer (i.e., the three controlled entities and their associated security attributes). The decision function $d_{print\text{-}sensitive}$ represents the permissible operation of printing a sensitive file.

As another example, an organization employing a confidentiality security policy may wish to prevent an employee from regrading an information object. In this case, the decision function $d_{regrade}$ could use a controlled-entity triple as its argument that includes the user, designated information object, and the information domain of the proposed regraded copy of the information object. It represents the permissible operation of copying an information object to a different information domain.

The values of the security attributes associated with each entity in the triple may be used by the decision function to determine whether or not the user is granted permission to execute the designated operation. The security policy may establish the constraints in the form of rules, conditions, or procedures, and guides the overall decision process.

Protection of a system is achieved through security policy decision functions (SPDFs) and security policy enforcement functions (SPEFs)[14] (generalizations of the access control decision functions and access control enforcement functions as described in the International Standards Organization (ISO) Security Frameworks [9-17]). The *separation* of SPDFs from SPEFs is the architectural approach specified by the DGSA and is intended to facilitate support for multiple security policies within an enterprise as described in Requirements 1 and 3 in Chapter 1. The enforcement, being separate from but dependent on the decision, may be carried out by an appropriate mechanism at any time after the decision has been made. There is the possibility that the "decision may expire" prior to use, in which case the decision must be re-evaluated.

---

14. A SPDF, as defined in the DGSA, is responsible for making all security policy decisions within an information system. A SPDF is employed in conjunction with a SPEF, forming the central protection mechanisms for an information system [5].

2.3 Information Domains

Having defined security policy and three types of primitive controlled entities, the primary component of information management, the information domain, can be defined. An information domain represents a set of shared information objects and a set of users constrained by a security policy, and establishes a logical view of information management, as illustrated in Figure 1. Associated with the information domains is a set of shared resources that are used in any associated computations.

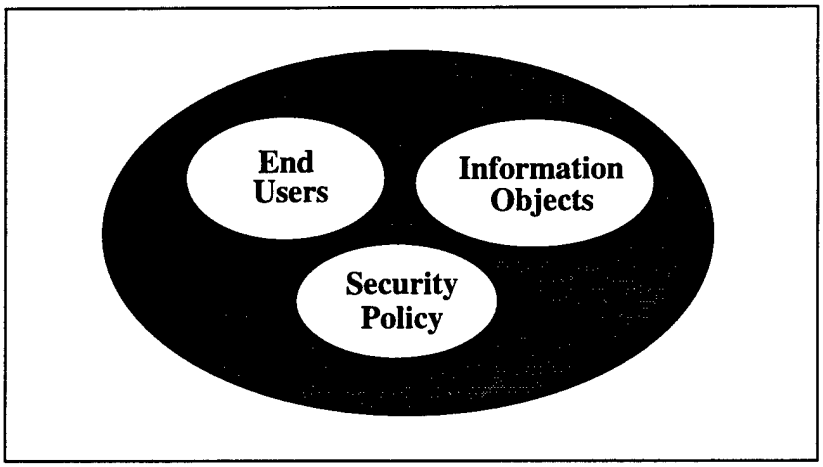

Figure 1. Conceptual View of an Information Domain

Enterprise operations and mission-oriented activities require routine interaction between groups of users and their information. The value of this information and, therefore, the required protection is determined by the group of users. The group establishes the criteria for membership, as well as the conditions for use of the information subject to any overriding policies. Controlled interaction between groups of users and information objects is achieved by the employment of information domains.

Three conditions must exist to successfully employ information domains: (1) a group must have a defined membership, (2) candidate information objects required by the group must be uniquely identified, and (3) the security policy regarding the protection of the information objects must be agreed to by the members of the group [5]. To explicitly define an information domain, the definition of security policy must first be extended.

> **Definition 2.7:** An *information domain security policy* $p \in \mathbf{P_D}$, where $\mathbf{P_D} \subset \mathbf{P}$ is the universal set of information domain security policies, is a statement of the criteria for membership of users in an information domain and the required

protection, including conditions of use, for the information objects in the domain.[15]

Using Definitions 2.2, 2.3, and 2.7, a formal definition of information domain can be constructed.

Definition 2.8: An *information domain* $d \in \mathbf{D}$, where **D** represents the universal set of all information domains, consists of a set $O \subseteq \mathbf{O}$ of uniquely identified information objects, a set $U \subseteq \mathbf{U}$ of users, and a domain security policy $p \in P_D$. At time t it is denoted $d_t = \langle O, U, p \rangle$.

With these definitions of information objects, users, information system resources, and information domains, two axioms[16] describing relationships between controlled entities and information domains can be stated.

Axiom 1: An information object cannot reside in two different information domains simultaneously.[17]

Every information object is associated with a domain identifier attribute and an object name attribute. The value of the object name attribute must be unique within that domain. Information domains are independent of one another and as a consequence of Axiom 1, there are no nested relationships (i.e., subset or superset relationships) between domains. The sensitivity of information objects[18] is determined by the security policy. The policy associated with one domain *need not* have any effect on the sensitivity of information objects in another domain. That is, membership in one information domain with a set of privileges in that domain does not automatically provide privileges in other domains of the same or differing sensitivities. For example, a user having authorized access to an information domain d_S containing information objects of sensitivity SECRET, cannot automatically gain access to another information domain d_C with information objects of sensitivity CONFIDENTIAL.[19] Likewise, access to objects in d_S does not imply access to information objects in d_S', another information domain with objects of sensitivity SECRET. Access to information objects per the security policy is granted because of explicit user membership in the information domain.

Axiom 2: A user can be a member of multiple information domains simultaneously.[20]

---

15. And by extension their associated information resources.

16. Axioms represent constraints of the DGSA as specified in [5].

17. Using mathematical terminology, Axiom 1 states that **D** induces a partition over **O** at time t.

18. If the sensitivity can be determined.

19. This is in contrast to the usual assumption where use is permitted if clearance of the user is greater than the classification of the information (and assuming "need to know").

Axiom 1 together with Axiom 2 implies that in different information domains different objects may be required to represent the (same) user. In general, there are no restrictions on information domain membership other than the criteria established by the appropriate information domain security policy. Information domains provide a mission-oriented view of information management and control, providing a framework in which shared information under a single security policy can be used to carry out a mission. In contrast to "domains" that are composed of information systems or networks of information systems, information domains are not bounded by physical systems or networks of systems. In fact, information domains are independent of the systems where the actual processing, transfer, or storage occurs. Instead, information domains are bounded exclusively by the presence of a set of users, a set of identifiable information objects, and a security policy. An information domain is therefore assumed to be supported on any physical system that can meet the protection requirements specified in the security policy [5].

2.3.1 Intra-Domain Constraints

Each information domain must be separately identifiable. This identification provides certain important information about the types of controlled entities that comprise the information domain as well as the types of activities that may occur within the domain. There are two intra-domain constraints[21] that must hold for an object in an information domain.

> **Constraint 1:** Information objects within an information domain must have the same set of policy-specified security attributes and the same security attribute values.

Constraint 1 implies that there is no security-relevant distinction made between information objects in an information domain. For example, information objects in some domain may have a security attribute representing *sensitivity*. The instantiated attribute value for each information object must be the same (e.g., COMPANY CONFIDENTIAL). Thus, all information objects in the information domain have the same attributes, and have the same value for each attribute.

> **Constraint 2:** Users within an information domain must have the same set of policy-specified security attributes but may have different security attribute values.

---

20. Again, employing mathematical terminology, Axiom 2 states that the universal set of users, **U**, need *not* induce a partition over the universal set of information domains, **D**. It is likely that a user will be part of numerous groups of users sharing information under different security policies.

21. Constraints may be derived from axioms and are stated explicitly in [5].

Constraint 2 implies that some users in an information domain may have different security privileges than other users in the domain in accordance with the domain security policy.[22] For example, users in an information domain might have a security attribute representing selected access privileges. User *u* might not be granted REGRADE permission while *u'* is granted this permission. However, the privileges for each user extend across all information objects in the information domain as required by Constraint 1.[23]

2.3.2 Inter-Domain Constraints

An enterprise can be characterized by its diverse and changing missions. The establishment of new missions, mission relationships, and organizations are the types of events that may trigger requirements to share and/or transfer[24] information objects between domains [5]. Information sharing and transfer operations must be accomplished under strict controls as articulated through information domain security policies established by the appropriate domain authorities, e.g., releasability constraints. There is a single security policy per domain and this policy must address all protection requirements for the domain to include the establishment of the rules, conditions, and procedures for information sharing within and transfer between domains. We will discuss the concepts of sharing and transfer independently and then describe the relationship between the two concepts.

Axiom 1 significantly restricts the type of information object sharing that can occur. The DGSA describes two possible methods for sharing that do not violate Axiom 1. The first method, and arguably the most simple, states that new users can be accepted into an existing information domain and be granted appropriate privileges. The second method is somewhat more restrictive and can be employed if more protection is required. That is, if there is a need

---

22. Conceptually, entities and attributes may be considered to be a relation in a relational database. Each information domain would have a unique database schema (as part of its relation) defining the set of security attributes required by the domain security policy. Each user could be viewed as a tuple in the relation with instantiated attribute values. Thus, while each user has the same policy-specified attributes (defined by the schema), attribute values can vary from user to user. This view of protection emphasizes identity-based and role-based protection, rather than protection based on the values of security attributes of information objects to be operated on.

23. This constraint represents a **significant** difference in the way objects are viewed in DGSA as compared with Multi-level Security (MLS) — an object cannot simultaneously be in both secret and top-secret domains. In many respects, the implementation is substantially simplified since only information flow into and out of information domains, not flow within a domain, need be mediated. User attributes take on a larger significance because they control whether or not a user may perform an operation on an object.

24. The DGSA defines a transfer operation as either a move operation or a copy operation. According to the DGSA, an information object is relabeled when it is moved or copied from one information domain to another, and it assumes the attribute values of the new domain.

to share some but not all information objects in an information domain, a new domain can be created with a particular set of information objects (to be shared) and a designated set of users.

Information transfer between domains presents new challenges that must be addressed by a set of specific rules that are a key part of the information domain security policy.

> **Definition 2.9:** An *information domain import-export policy* is that part of an information domain security policy that establishes the rules, conditions, and procedures for the transfer of information objects with other domains.

Import-export policy relates the security policies of distinct information domains and also establishes constraints for information flow between domains. This policy reflects the agreement between participating domains regarding the inter-domain transfer of information objects; both domains must allow the transfer. Import-export policy addresses user security attributes, such as *release*, as well as specific protections, such as auditing and identification, that are required when information objects are transferred between domains.

> **Constraint 3:** The user responsible for an inter-domain transfer of information objects must be a member of both the source and the destination domains and must be permitted to make the transfer by the import-export policies of each domain.

3. Information System Dependent DGSA Concepts

3.1 Information Systems

To complete the set of information management components, the physical systems where information processing, transmission, and storage occur must be discussed. These systems are in general referred to as information systems and are characterized by their constituent parts. Specifically, the DGSA views an information system at the architectural level consisting of a set of local subscriber environments connected to one another by communications networks.[25] Building on the previous definitions, the components of an information system are considered controlled information system resources. The remainder of this section provides descriptions and formal definitions for the individual components of an information system as presented in the DGSA.

3.1.1 Local Subscriber Environment

The local subscriber environment (LSE) consists of all information processing, transfer, and storage devices under user or organization control [5]. These devices can be categorized as end systems, relay systems, and local communications systems.[26] The principal component of the LSE is the end system where user applications and data are processed and stored.

> **Definition 3.1:** An *end system* es $\in$ **ES**, where **ES** represents the universal set of all end systems, is an information processing system to include processor and input/output devices (e.g., workstation, personal computer, server, minicomputer, mainframe, disk drive, printer, telephone) directly accessible by users.

Within the LSE, local communications systems (e.g., local area networks) provide the connectivity between end systems.

---

25. Requirements 2 and 4.

26. The DGSA distinguishes between end systems and relay systems. An end system provides traditional information processing capability whereas a relay system provides limited functionality related to information transfer and is employed primarily in support of a transfer system (described later in this section). In reality, end system functions and relay system functions can exist on the same (hardware) platform. For purposes of this paper, references to end systems are taken to be inclusive of relay systems as well.

Definition 3.2: A *local communications system* lcs $\in$ **LCS**, where **LCS** represents the universal set of all local communications systems, is a set of communication devices (e.g., ring, bus, twisted pair, coaxial cable, fiber-optic cable) under the direct (physical) control of a local subscriber environment.

Mission requirements often dictate that users in different LSEs communicate with one another across a network of end systems. Specialized information processing systems, or relay systems, operating as part of wide-area communications networks provide this capability.

Definition 3.3: A *relay system* rs $\in$ **RS**, where **RS** represents the universal set of all relay systems, is an information processing system (e.g., multiplexor, router, switch, cellular node, message transfer agent) not directly accessible by users[27] that manages transfers of information between a public network and a local communications system.

Using Definitions 3.1, 3.2, and 3.3, a formal definition of a local subscriber environment can be constructed.

Definition 3.4: A *local subscriber environment* lse $\in$ **LSE**, where **LSE** represents the universal set of all local subscriber environments, is a triple $\langle ESL,RLS,LCSL \rangle$, where $ESL \subseteq$ **ES** is an ordered list of end systems[28], $RSL \subseteq$ **RS** is an ordered list of relay systems, and $LCSL \subseteq$ **LCS** is an ordered list of local communications systems.

With these definitions, it is possible to state an axiom describing the relationship between information system components and a local subscriber environment.

Axiom 3: An end system, a relay system, or a local communications system cannot reside in two different local subscriber environments simultaneously.[29]

Axiom 3 states that LSE boundaries do not overlap and that each end system, relay system, and local communications system must belong to a distinct local subscriber environment.

3.1.2 Communications Network and Transfer System

Having defined the components of an LSE, we now describe the communication infrastructure that connects the end systems together over a distributed network. The successful linkage of multiple end systems can be attributed to the communications networks and transfer systems.

---

27. Users do not observe or modify information objects on a relay system.

28. An ordered list is required for the EDS Model introduced in Section 3.3.

29. Using mathematical terminology, Axiom 3 states that **LSE** induce a partition over **ES**, **RS**, and **LCS**.

Communication devices outside the direct local control of LSEs are part of a larger communications network used to connect LSEs. Communications networks, in general, can be implemented using private links or public common carrier links. Private networks have both ends of the communication links terminating in an LSE without passing through intermediate network nodes. Network communications, in this situation, can be protected with complete traffic flow security. In contrast, common carrier networks have commercial control zones that provide intermediate network nodes used for routing network traffic to its final destination. Since complete address information is needed for common carrier network implementations, traffic flow security is limited [5].

The transfer system is a logical grouping of communication protocols integrated into end systems, relay systems, local communications systems, and communications networks. Figure 2 illustrates the conceptual view of an information system with its fundamental components [5].

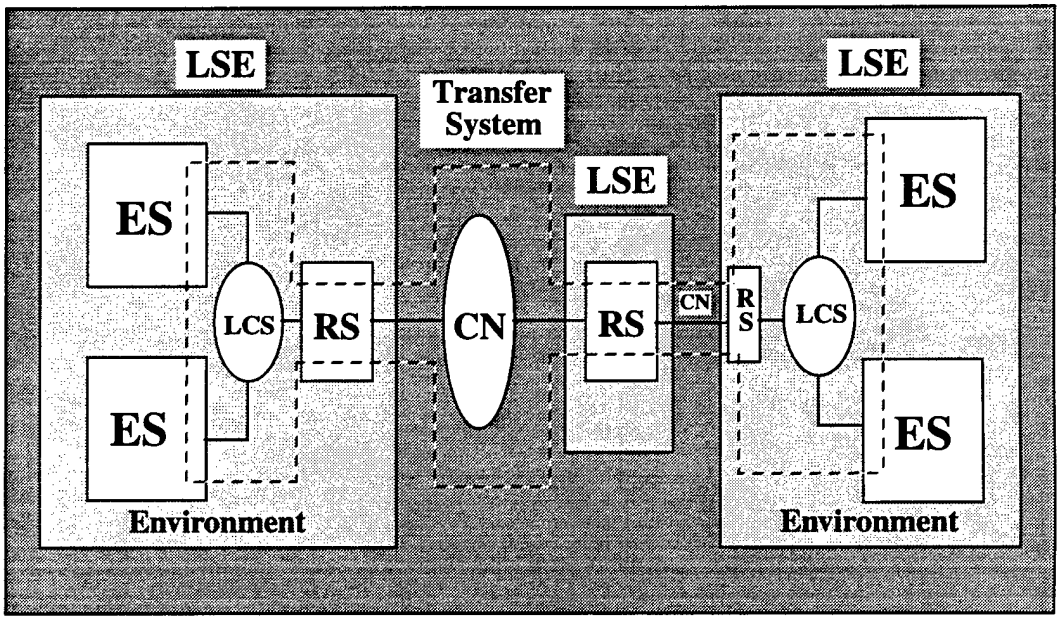

Figure 2. Conceptual View of an Information System [5]

3.2 Information Domain and System Relationships

Information domains have been described in Section 2.3 as providing a mission-oriented, logical view of information management. The discussion on LSEs and communications networks described the physical environment where information processing, transfer, and storage occur. From a system implementation standpoint, it is important to be able to relate the logical

concept of an information domain to the physical information system environment. To facilitate this task, some of the axioms defined in Section 2.3 are extended from the logical information domain construct to the physical system construct, and a taxonomy of relationships between information domains and end systems is defined.

3.2.1 Information Domain Projection onto End Systems

While it is convenient to describe the logical construct of an information domain as a collection of information objects, users, and a domain security policy, in reality the controlled entities comprising an information domain are distributed across a set of one or more end systems in their local environments. It is also likely that these end systems are interconnected within a larger network of information systems. The term *projected* is used to describe the actual implementation of an information domain on a specific end system or end systems. For example, if information domain d is projected onto three end systems es_1, es_2, and es_3, all three end systems are processing, transferring, and/or storing information objects from d.

Using the concept of domain projection onto end systems, the following axioms appropriately extend and amplify Axiom 1 and Axiom 2.

Axiom 4: An information object may only exist on one end system at a time.[30]

Axiom 4 implies that each information object is unique. Each object has a domain unique identifier. The contents of an object container (e.g., a file) could be identical to the contents of another object container, but the container is labeled uniquely.

Axiom 5: Users may operate on more than one end system simultaneously.[31]

In general, there are no restrictions on multiple end system operations other than those restrictions and controls dictated by the information domain security policy as implemented through specific information system security policies.

3.2.2 Taxonomy of Domain and System Relationships

Given the previous definitions of information domain and information system, a taxonomy of relationships can be defined with respect to domain-system projection. As illustrated in Figure 3, the taxonomy includes four generalized cases of information domains projected onto

---

30. Using mathematical terminology, Axiom 4 states that the set of information objects, O, within an information domain, d, induces a partition over the set of end systems, ES, which host d.

31. Again, employing more formal terminology, Axiom 5 states that the set of users, U, within an information domain, d, need not induce a partition over the set of end systems, ES, which host d.

end systems: single-domain centralized system, single-domain distributed system, multi-domain centralized system, and multi-domain distributed system.

The *single-domain centralized system* is the simplest domain-system relationship representing a single end system processing information objects from a single information domain with no connectivity to other end systems. A single domain requires no underlying mechanisms to enforce domain separation as there are no competing information domains sharing the resources of the end system.

The *single-domain distributed system* is a one-to-many domain-system relationship representing a network of end systems processing information objects from a single information domain. A single information domain, as in the first case, requires no underlying mechanism support to enforce separation on a particular end system. Since the DGSA provides a set of constraints which determines the class of acceptable implementations, it is possible that in a heterogeneous environment with multiple end systems, different implementations of protection will be used on different end systems. Thus interoperability of protection systems must be considered. In addition, multiple end systems dictate the use of protection mechanisms for information objects in transit between end systems. The specific details on protection of information in transit are addressed in Section 3.3.2.

The *multi-domain centralized system* adds complexity by increasing the number of distinct information domains operating on an end system. The many-to-one domain-system relationship represents a single end system processing information objects from multiple information domains with no connectivity to other end systems. The introduction of multiple information domains requires appropriate protection mechanism support on the end system *to ensure domain separation and enforcement of the domain security policies*. Each security policy could potentially require a set of security mechanism which differ from those used in supporting other domains. The specific details of information domain separation and mechanism support on end systems are discussed in Section 3.3.2.

The *multi-domain distributed system* is the most general domain-system relationship. The many-to-many relationship represents a network of end systems simultaneously processing information objects from multiple information domains. As in the previous case, multiple information domains require appropriate protection mechanisms on the end systems to ensure domain separation and security policy enforcement. In addition, the employment of multiple end systems in a distributed network requires the use of appropriate protection mechanisms for information objects in transit, as well as interoperability of heterogeneous protection mechanisms across the distributed domain.

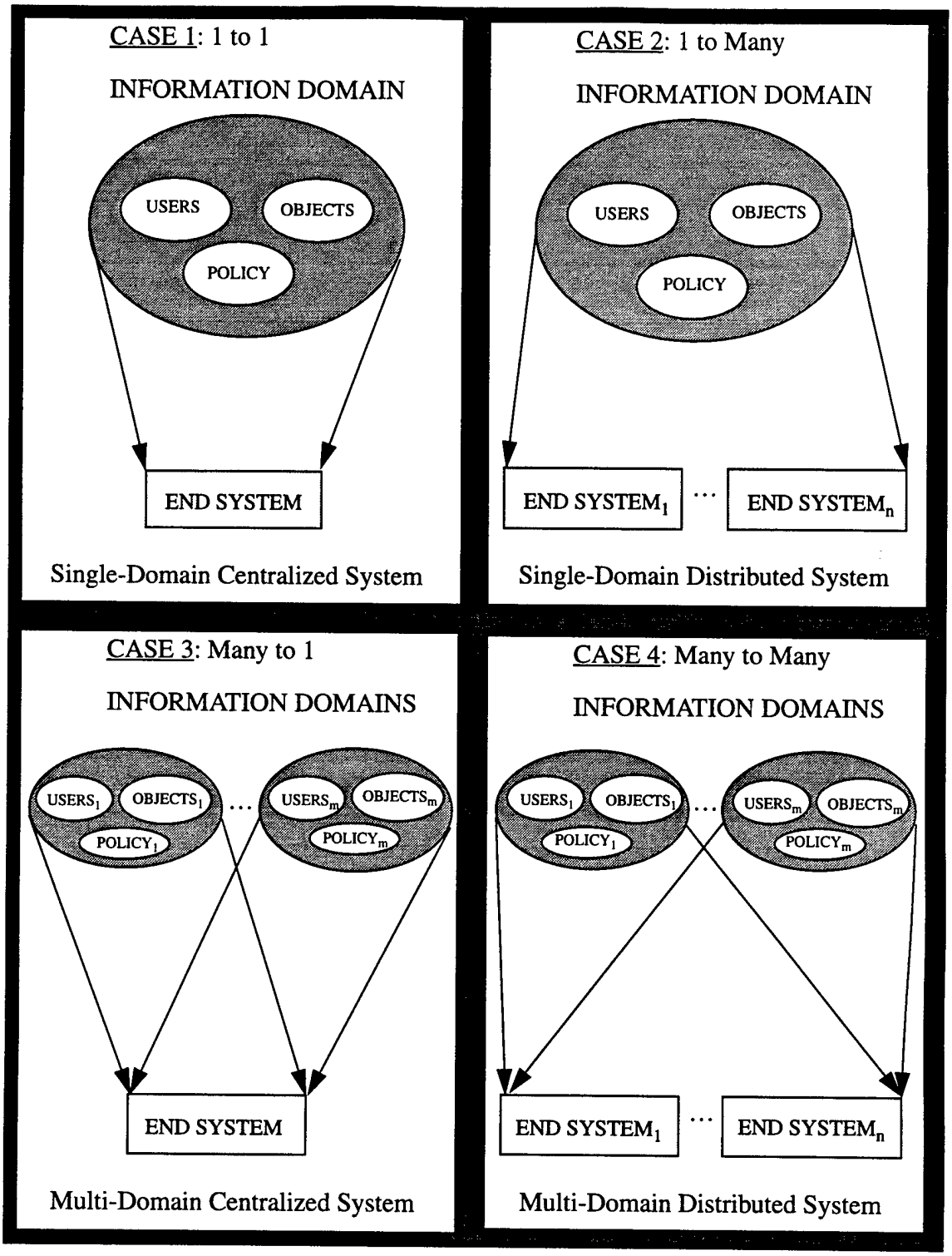

Figure 3. Taxonomy of Information Domain-System Relationships

3.3 Entity-Domain-System Model

The previous section defined the individual components of information management. Given the components of security policy, controlled entities, information domain, and information system, it is possible to define an abstract model that unifies these components and facilitates understanding of their interaction. This section develops such a model and establishes the foundation for discussing the fundamental security concepts articulated in the DGSA.

The Entity-Domain-System (EDS) Model* is an abstract representation of the DGSA and provides a framework to describe the management, administration, and implementation issues associated with the transition to the goal security architecture. In general, the EDS Model represents the operation of information domains on a set of end systems in their local environments. The following sections provide a detailed description and derivation of the model.

3.3.1 Defining the Three-Dimensional Model View

The EDS Model provides a three-dimensional view of information management, with each dimension representing one of the key components previously defined. Using a three-dimensional coordinate system as a frame of reference, at any given instant of time the x, y, and z axes represent information domains, controlled entities, and end systems, respectively.[32] Thus, each entry on the x-axis represents a specific information domain d_i, each entry on the z-axis represents a specific end system es_j, and each entry on the y-axis represents a controlled entity. The entries in the model indicate in which domains and on which end systems the various entities exist.

An information domain security policy is implemented locally on an end system by utilizing controlled information objects that represent policy. As stated previously, a domain security policy must eventually be translated into an information system security policy in order to enforce the domain security policy on an end system. As such, the domain security policy becomes a policy information object, or set of objects, within a specified information domain when projected onto an end system.[33] Figure 4 illustrates the three-dimensional view of the EDS Model. Specific derivations of each component of the model are provided below.

---

32. While the EDS Model uses the term *three-dimensional coordinate system* to model the components of information management, each entry on the x-axis, y-axis, and z-axis is a specific entity represented as a volume and should not be interpreted in the mathematical context of actual ordered coordinates along axes.

33. Information domains containing security policy-related information objects are discussed in greater detail in Section 3.4.

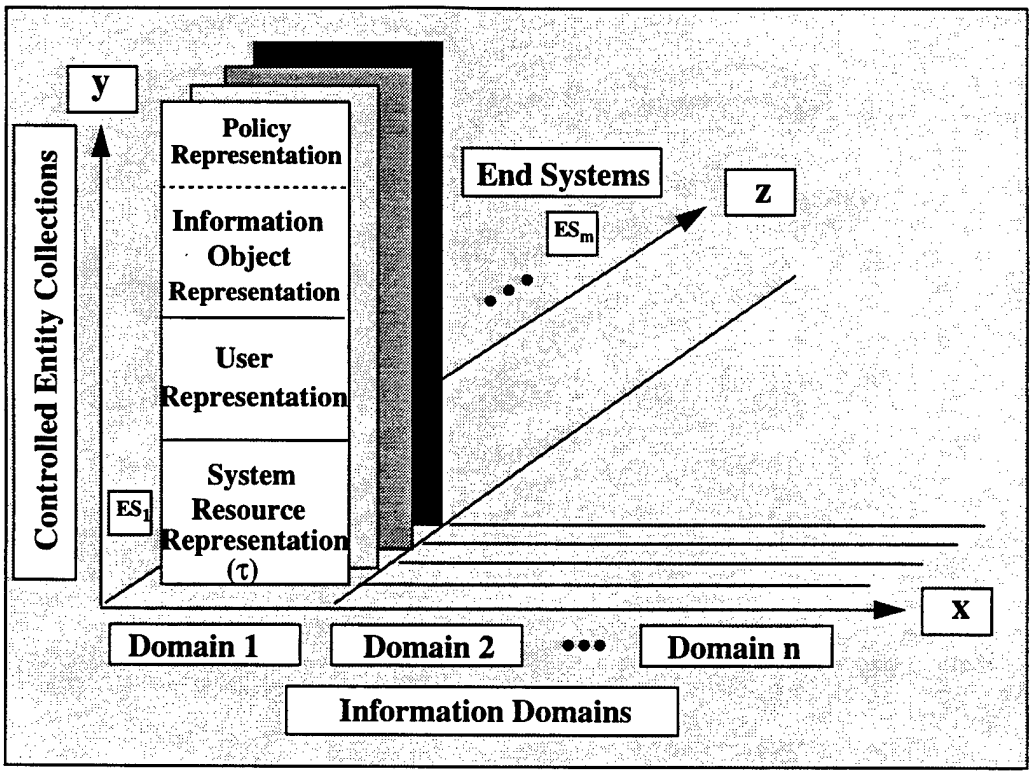

Figure 4. The Entity-Domain-System Model at time τ

3.3.2 Components of the EDS Model

In deriving the components of the EDS Model, we must first address the set of controlled entities defined by the *y*-axis, or the set of representations of controlled entities associated with a specific end system and information domain. (Note that these representations may not be identical for every instantiation of an information domain). To accomplish this task, several additional definitions are needed to distinguish the respective types of controlled entities within the model.

3.3.2.1 Controlled Entity Collection

Having defined both the logical construct of an information domain and its projection onto a set of one or more end systems, a domain can be described, in further detail, in terms of its physical distribution across end systems. Note that the totality of controlled entities (i.e., users, information objects, and resources) in an information domain must be distinguished from the various subsets of those controlled entities operating on specific end systems. That is, each end system servicing an information domain has a subset of controlled entities operating

in its local environment. Recall from Definition 2.8 that an information domain $\langle O, U, p \rangle$ consists of a set $O \subseteq \mathbf{O}$ of information objects, a set $U \subseteq \mathbf{U}$ of users, and a domain security policy $p \in P_D$. Thus, the projection of an information domain onto a set of one or more end systems produces a collection of controlled entities[34].

> **Definition 3.5:** A *controlled entity collection*$\ast$ $ce_{(x,z)}(t) \in \mathbf{CE}$, where **CE** is the universal set of all controlled entity collections, is a set of controlled entities at time t obtained by holding the information domain on the x-axis and the end system on the z-axis constant, and taking the vertical projection along the y-axis.

An information domain, then, is specified by a set of controlled entity collections. Let d represent an information domain projected over a set of end systems $ES = \{es_1,...,es_q\}$ (q>0). Then d is defined by the controlled entity collection set $\{ce_{(d,es(1))},...,ce_{(d,es(q))}\}$ as

$$d(t) = \langle (O_1 \cup ... \cup O_q), (U_1 \cup ... \cup U_q), p \rangle$$

where each $ce_{(d,es(i))}(t) = (O_i \cup U_i \cup R_i \cup \{\pi_i\})$ with $O_i \subseteq \mathbf{O}$[35], $U_i \subseteq \mathbf{U}$, $R_i \subseteq \mathbf{R}$[36], and π_i the implementation of p on es_i as a unique set of information objects. Note that the π_i need not be the same, and may even require a different set of implemented services, but they must support p. The distribution of controlled entities from an information domain must be accomplished in accordance with Axioms 4 and 5.

In the EDS Model, the controlled entity collection is the conceptual structure that unites the logical information domain with the physical information system. Given Definition 3.2, we can define an axiom describing the relationship between information system resources and controlled entity collections.

> **Axiom 6:** An information system resource can support (be a member of) more than one controlled entity collection, but only in a time-multiplexed manner.[37]

---

34. A *controlled entity collection*, as defined in this paper, has no relation to the traditional concept of a collection as a data structure in a computing system. That is, while a controlled entity collection can be considered a data structure for modeling purposes, there is no implied order or entry/exit criteria for elements in the collection. A controlled entity collection is dynamic in nature, and represents a group of controlled entities associated with one another because of their membership in a particular information domain and the specific operating environment where information processing, transfer, and storage occur. Thus, the controlled entity collection changes over time with entities being added and removed as required.

35. Due to the dynamic nature of information domains, the projection of an information domain onto a set of end systems may result in some controlled entity collections containing no information objects at a particular instant of time. The same situation can occur with users.

36. The distinction between R_i and es_i is noteworthy. In the EDS Model, an end system, *es*, contains a set of information system resources, R_{ES}, that supports its local operating environment.

Thus, at any point in time, a subset of information system resources $R_{es} = \{r_1,...,r_l\}$ ($l \geq 0$) is dedicated to supporting a particular user (or collection of users who share a resource), in an information domain, on a particular end system.

3.3.2.2 Information Domain and End System Views

Using Definition 3.2, two distinct views of information management can be defined. Let $D = \{d_1,...,d_v\}$ ($v>1$) be a set of information domains that is projected over a set of end systems $ES = \{es_1,...,es_q\}$ ($q>1$).

> **Definition 3.6:** An *information domain view* of information management at time t is defined by a set of controlled entity collections CE obtained by holding information domain d (on the x-axis) constant (i.e., selecting a particular information domain), and taking the vertical projection along the y-z plane. It is the union of controlled entity collections across information domain d.

$$CE_d = \bigcup_{1 \leq j \leq q} ce_{(d,es_j)}(t)$$

Conversely, a complementary view can be defined by fixing the end system.

> **Definition 3.7:** An *end system view* of information management at time t is defined by a set of controlled entity collections CE obtained by holding end system es (on the z-axis) constant (i.e., selecting a particular end system), and taking the vertical projection along the x-y plane It is the union of controlled entity collections across end system es.

$$CE_{es} = \bigcup_{1 \leq i \leq v} ce_{(d_i,es)}(t)$$

3.3.3 Interpreting the EDS Model

The EDS Model is fully generalizable and can be used at various levels of abstraction. For example, the three-dimensional model can represent an enterprise as large as the DoD or an organization as small as a private-sector company. For the former, the model represents all information domains and end systems within the DoD. For the latter, the model represents only those information domains and end systems owned by the respective company. It is also possible to view the EDS Model at the most abstract level where the dimension along the x-axis represents the universal set of all information domains **D**, and the z-axis represents the universal set of all end systems **ES**.

---

37. Using mathematical terminology, Axiom 6 states that **CE** induces a partition over $\mathbf{R}_t$ at time t.

3.4 Relating the DGSA Concepts to the Model

Section 3.3 provided detailed descriptions of the components of information management and an abstract model to facilitate understanding of the complex interactions between controlled entities, information domains, and information systems. We can now describe the fundamental security concepts articulated in the DGSA and develop an interpretation of those concepts with respect to the abstract model. The taxonomy of information domain-system relationships (Figure 3) is used to describe the types of generic configurations that are useful in highlighting certain security concepts.

3.4.1 Security Contexts

The diversity of missions and information protection requirements results in the proliferation of information domains, each with its own security policy and protection requirements.[38] To support multiple information domains on a single end system, as illustrated in the multi-domain centralized system example (Figure 3, Case 3), a protection strategy must be developed that effectively satisfies the requirements specified by each of the individual domain security policies. The strategy employed by the DGSA to support multiple information domains on an end system is strict isolation [5].

> **Definition 3.8:** *Strict isolation* is the absolute separation of a set of information domains $\{d_i,...d_n\}$, n > 1, *and* their associated controlled entities from other domains on a specific end system at any instant of time.

The end system, through its underlying hardware features and operating system functions, must provide appropriate security mechanisms to enforce the separation between domains in such a manner as to satisfy the requirements of each information domain security policy.[39] The DGSA mandates that the strict isolation between information domains be enforced as a default condition, unless an explicit relationship between domains is defined by an information domain security policy through import-export policies.

In implementing strict isolation, information objects must be confined to their information domains. This is accomplished, in part, via the security context. Recalling that an information domain is defined as $\langle O, U, p \rangle$, a formal definition for security context can be constructed.

---

38. While this statement implies that there will be many unique information domain security policies, in reality, a number of domain security policies may be very similar. Thus, when new information domains are created, there could be significant potential for re-use of existing security policies—only changing the policy where necessary to meet the specific protection requirements of the new domain. Each domain might have a different set of users associated with the policy.

39. This is assumed to be possible.

Definition 3.9: Let $d \in \mathbf{D}$ be an information domain and $es \in \mathbf{ES}$ be an end system. A *security context* $sc_{(u,d,es)}(t) \subseteq ce_{(d,es)}(t) = O \cup U \cup R \cup \{\pi\}$ supporting $u \in U$ (where R is a set of resources and π is the implementation of p on es) is a set of controlled entities $O' \cup \{u\} \cup R' \cup \{\pi\}$, $O' \subseteq O$ and $R' \subseteq R$, forming an operating environment for u in domain d on end system es at time t.

A security context $sc_{(u,d,es)}(t)$ provides services and/or facilities to a specific environment for u. It is closely related to the concept of a user or system process space implemented by an operating system and selected hardware features. In essence, a security context is a collection of all data, programs, and system resources (e.g., hardware, system software, user application software, and information) necessary to support a particular user or system function operating in a particular information domain on a specific *es* in accordance with a system specific domain security policy.[40] Thus, a user must have a specific security context established for each information domain where processing is required. An end system may maintain multiple security contexts depending on the number of information domains and users supported. Figure 5 illustrates a simplified conceptual view of a security context.[41]

Given an end system environment supporting multiple information domains and users, the individual security contexts must be isolated from one another. This isolation is accomplished on the end systems through hardware, firmware, and operating system mechanisms.[42] The operating system must maintain all essential information required to enforce security context separation. Thus, a security context is established for user u, operating in information domain d, on end system, *es*.

3.4.2 Security Management

Security management is concerned with the management of security policies, security services, security mechanisms, mechanism support, and transfer system security. Within the information system environment, there must be appropriate supporting infrastructure to effectively manage the information domains that are resident on end systems. Security management

---

40. The DGSA also includes *security doctrine* as part of a security context. Security doctrine addresses the specific conditions of use for a particular component, facility, or system. The specification of conditions of use within a specific environment is intended to complement the protection provided by the hardware, firmware, and software mechanisms as part of the original product design [5]. The topic of security doctrine is beyond the scope of this paper.

41. Figure 5 illustrates that multiple security contexts can exist on an end system obtaining users, objects, and policy from the same information domain. Nevertheless, each security context is distinct and maintains strict isolation from other security contexts.

42. The approach described in the DGSA calls for the isolation of security contexts through the employment of a *separation kernel* similar to that defined by Rushby[21].

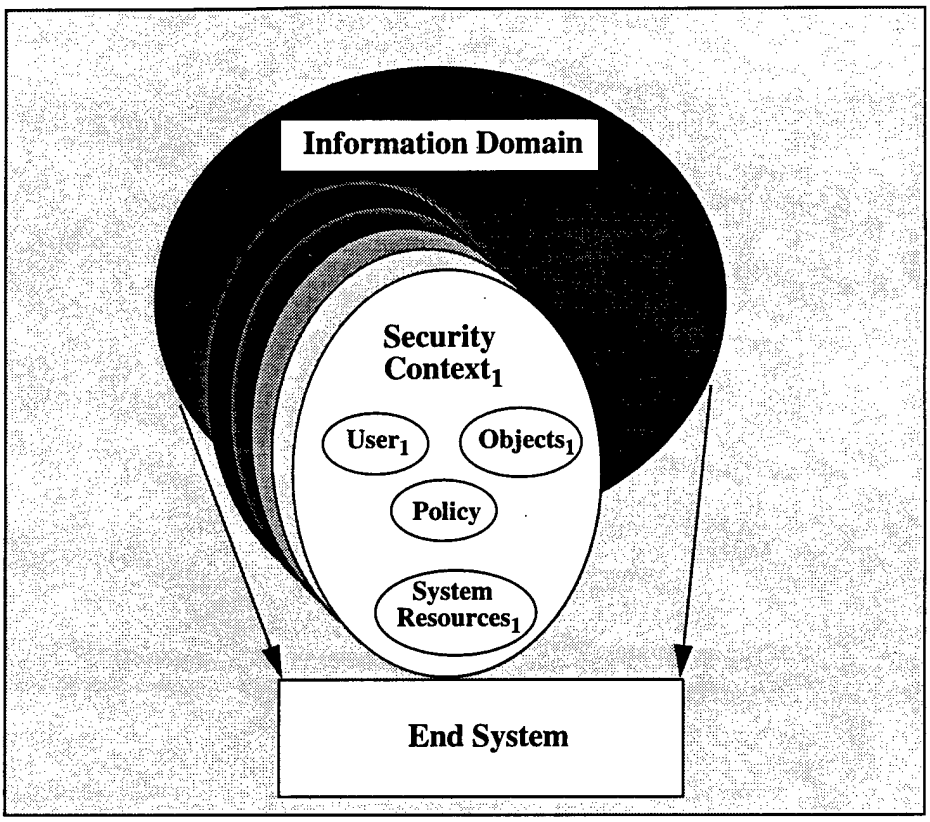

Figure 5. Simplified Conceptual View of Security Contexts

provides the security services necessary for the protection of controlled entities on an end system in accordance with applicable information domain security policies.

3.4.2.1 Security Management Information Domains

The security management information necessary to implement the supporting infrastructure must be separated from the controlled entities within the supported information domains. In general, this separation is accomplished by placing all critical security management information in distinct information domains. That is, security management information is maintained as sets of controlled entities in security management information domains.

A security management information domain consists of the same components as a general purpose information domain, including a set of information objects, a set of users, and a domain security policy. The information objects consist of security management information (data and programs) necessary to provide appropriate protection for the domain. Each set of security management objects supporting a particular information domain is contained in a log-

ical repository called a *security management information base (SMIB).*[43] Users within the security management information domain are typically privileged users such as systems administrators or system security officers. The domain security policy provides a statement of the criteria for membership of users in the security management information domain and the required protection for the information objects in the domain. Given the general description of a security management information domain, it is possible to construct a more formal definition.

> **Definition 3.10:** A *security management information domain* $d_M = \langle O_M, U_M, p_M \rangle$ comprises a set of uniquely identified information objects $O_M \subseteq \mathbf{O}$ grouped into security management information bases (SMIBs), a set of privileged users $U_M \subseteq \mathbf{U}$, and a management information domain security policy $p_M \in \mathbf{P}$.

Figure 6 illustrates a conceptual view of a security management information domain with its constituent components. In addition to security management information domains, there are end system information domains as described in Section 3.4.2.2.

3.4.2.2 End System Information Domains

In general, security management is a function of the end system providing support for the information domains. End systems that support multiple information domains must be capable of providing independent security management for each information domain as well as separate security management for shared system resources (e.g., security functions, services, mechanisms, devices, memory, registers) at the end system level. Thus, in addition to the domain security policies, there must be a separate security policy established for the end system that addresses the management of shared system resources.

> **Definition 3.11:** An *end system security policy* π_{SYS} is a security policy that specifies requirements for sharing of information system resources (e.g., security functions, services, mechanisms, devices, memory, registers) on an end system *es* in support of a set of information domains resident on *es*.

The end system security policy is separate and distinct from the security management information domain security policy and focuses solely on the sharing of information system resources. End system security policies must be controlled in the same manner as information domain security policies, and therefore, exist within an end system information domain. End

---

43. SMIBs supporting information domains contain information domain policy rules, user registration information, user authentication criteria (e.g., strength of mechanism required), user security attributes, and security service and security mechanism requirements for inter-domain information transfers. Programs in execution which operate on the SMIB are called *security management application processes (SMAP)* and are considered information system resources.

system information domains are used to control and manage system resources (e.g., login procedures, security management information domains, and multi-domain objects[44]).

In addition to the SMIBs supporting each information domain, each end system has at least one SMIB to control the shared system resources. Security management information bases supporting end systems typically contain end system security policy rules, management information for security services and mechanisms, and management information for supporting services and mechanisms (e.g., auditing, alarm reporting, key distribution, security contexts) [5].

The end system SMIB, along with the end system security policy, privileged users, and information system resources, becomes part of the controlled entity collection for a particular information domain and end system, augmenting the controlled entities previously accrued from the projection of the domain onto the end system. Thus, the controlled entity collection is the unifying structure that brings together the controlled entities from the (logical) information domains and the repository of (physical) information system resources.

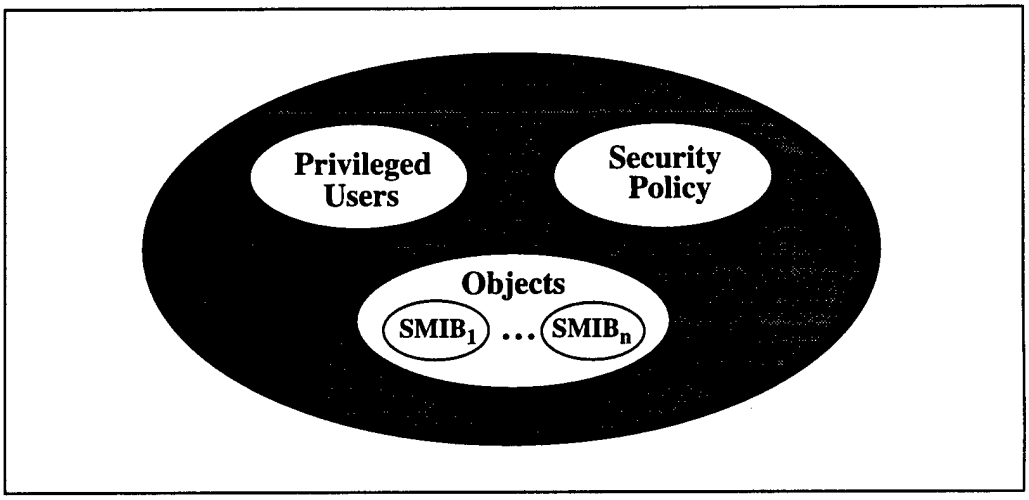

Figure 6. Conceptual View of a Security Management Information Domain

The information resource must be the principal focus for matters of assurance. Use of shared resources offers the possibility of covert timing and storage channels. Inappropriate sharing of an information resource appears to be the only way that the information from an information object can be transferred from one domain to another, assuming that the security policy for the information domain calls for strict isolation and this policy is properly enforced.

44. Multi-domain objects are special composite virtual information objects created from constituent objects from different information domains and are strictly limited in their use. Multi-domain objects are discussed in greater detail in Section 3.4.5.

Labeling of information with respect to the domain to which it belongs is usually necessary before that information can be sent to a shared resource.

3.4.2.3 Projecting Security Management Domains onto End Systems

Depending on operational requirements, the components of a security management information domain can be distributed across multiple end systems. Thus, as with information domains in general, security management information domains can be projected onto a set of end systems, producing a set of controlled entity collections for security management. A controlled entity collection represents the contents of the security management information domain resident on a particular end system. For example, the information objects of each *SMIB* (or portion thereof) supporting an information domain can be distributed across the end systems on which the domain is implemented. Privileged users from the security management information domain may also operate on different end systems. The controlled entities from the specific security management information domain become part of the controlled entity collection for the particular end system designated to receive the security management information as a result of the domain projection.

3.4.2.4 Taxonomy of Security Management Information Domains

There are several ways that a security management information domain can be employed to support general purpose information domains, and thus it is possible to define a taxonomy of relationships between supporting and supported domains. As illustrated in Figure 7, the taxonomy describes three general cases of support relationships between security management information domains and information domains (i.e., one-to-one, one-to-many, and embedded) [5].

The first case, *single-security-management-domain-to-single-information-domain*, describes the support relationship in which one security management information domain is dedicated to supporting one information domain. The security management information domain contains only one SMIB, for the supported information domain.

The second case, *single-security-management-domain-to-multiple-information-domains*, is an extension of Case 1, and describes the support relationship in which one management information domain supports a set of *n* information domains, $n > 1$. The security management information domain contains *n* SMIBs (i.e., one SMIB per information domain supported). The third and final case, *embedded-security-management-information-domain*, is a special case in which all of the security management information is embedded in the infor-

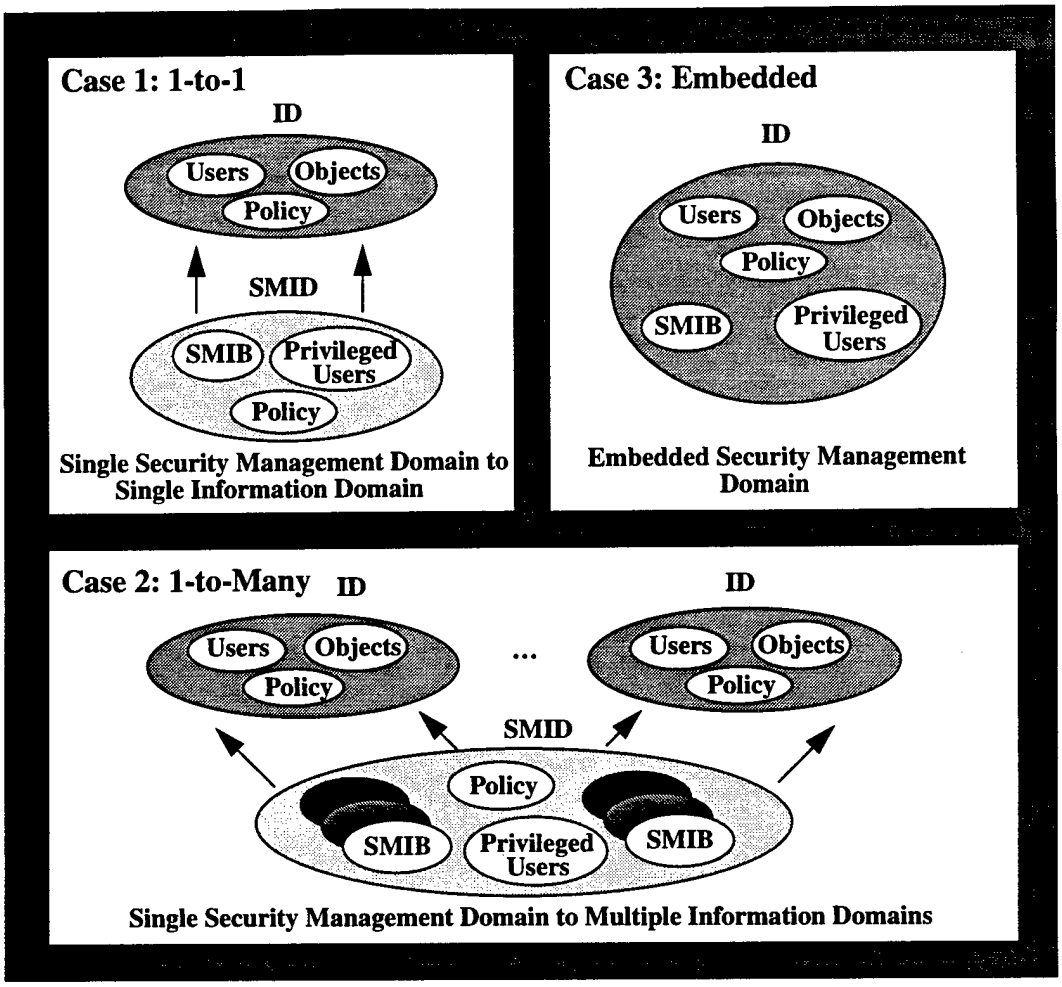

Figure 7. Security Management Domain Support Relationships

mation domain supported.[45] Again, as in Case 1, the information domain contains a single SMIB related to the supported information domain

3.4.3 Distributed Security Contexts

Complex enterprise missions often necessitate intra-domain communication within a set of distributed end systems. To support intra-domain communication on multiple end systems, the protection strategy of strict isolation must be extended to effectively satisfy the domain security policy on each of the end systems where the information domain operates. The

45. A security management information domain, for example, may contain its own security management information.

strategy employed by the DGSA to support information domain operation on multiple end systems is absolute protection [5].

> **Definition 3.12:** For a given information domain d projected onto a set of end systems $ES = \{es_1,...,es_q\}$, $q > 1$, *absolute protection* is the condition of achieving the minimum required strength of protection for d on each $es \in ES$.

Using Definition 3.12, we can formalize the concept of absolute protection for an information domain d and a set of end systems ES within an LSE. Let special function $protected_{(d,es)}$ describe the condition of achieving the minimum required strength of protection for d on a specific end system $es \in ES$. Then, absolute protection can be defined by the following expression:

$$protected_{(d,\,ES)} = protected_{(d,\,es_1)} \bullet \ldots \bullet protected_{(d,\,es_q)}$$

where • represents a logical *and* operator in boolean algebra.

Absolute protection requires that each end system supporting an information domain possess the minimum requisite strength of protection necessary to ensure the domain security policy is adequately enforced. Consistency of protection across end systems is the hallmark of absolute protection and is a function of the effectiveness and correctness of security mechanism implementation.[46] *Note that the definition is only valid within a LSE* — the LCS is assumed to be sufficiently protected.

The DGSA transfer system is intended to create an environment in which separate security contexts, operating from different controlled entity collections on physically distributed end systems, can communicate securely as if the security contexts resided on the same end system. The need for secure, distributed communications between end systems supporting the same information domain produced the requirement for distributed security contexts.

> **Definition 3.13:** Let $d \in \mathbf{D}$ be an information domain, $ES = \{es_1,...,es_q\} \subseteq \mathbf{ES}$ $(q > 1)$ be a set of end systems, and $CE = \{ce_{(d,es1)}(t),\ldots, ce_{(d,esq)}(t)\}$ be a set of controlled entity collections. A *distributed security context* $dsc_{(u,d,ES)}(t) \subseteq \cup CE = O \cup U \cup R \cup \{\pi\}$ supporting $u \in U$ is a set of controlled entities $O' \cup \{u\} \cup R' \cup \{\pi\}$, $O' \subseteq O$ and $R' \subseteq R$, forming an operating environment for u in domain d on the end systems in ES at time t.

---

46. While the DGSA mandates strength of mechanisms be consistent across end systems, the goal architecture allows maximum flexibility in actual design and implementation of security mechanisms. There is also no inherent restriction on providing additional protection for an information domain, given that the minimum required strength of protection (on the end system) has been achieved in accordance with the domain security policy.

A distributed security context can be interpreted as joining end system security contexts that have been established for a specific user in support of the same information domain, thus giving the user use of information domain objects that reside on remote (non-local) end systems using the information system resources of the local and remote systems. It is possible to formally define a distributed security context over end systems *es* and *es'* as

$$dsc_{(u,\,d,\,es)}(t) = sc_{(u,\,d,\,es)}(t)|X|sc_{(u,\,d,\,es')}(t)$$

where |X| is a special infix operator representing the *joining* of two security contexts. Generalizing the above equation, an n-way distributed security context[47] can be expressed as

$$dsc_{(u,\,d,\,ES)}(t) = sc_{(u,\,d,\,es_1)}(t)|X|\ldots|X|sc_{(u,\,d,\,es_q)}(t)$$

The DGSA defines two types of distributed security contexts, staged delivery and interactive. In a *staged delivery distributed security context*, there exist *n* relay systems between the originating end system and the destination end system. Information is sent in its entirety from the originating end system, in turn, through the set of *n* relay systems and (n-1) CNs until the destination end system is reached. Since, by assumption, there is an absence of security services throughout the set of CNs (other than availability), the security context from the originating end system must be transmitted securely across the network and then reconstructed on the destination end system. Based on the information domain security policy and its projection on end systems π_i and π_j, the protection for the information in transfer can be provided by cryptographic wrapping or other available techniques. A common example of staged delivery communications is an electronic mail application.

The alternative to a staged delivery distributed security context is an *interactive distributed security context*. Interactive distributed security contexts develop a secure virtual channel between end systems. This secure virtual channel is established using a set of security mechanisms called a security association.[48] The security policy of the information domain may have a distinct sub-policy relating to the *transfer policy* which must be enforced during transfer of information from end system to end system.

---

47. While a distributed security context can be *n*-way, the current state of information system security technology and practical considerations limit the formation of distributed security contexts to pair-wise associations.

48. See for example the security associations specified in the IPSEC. Note that these security associations are host to host and not application to application.

Definition 3.14: A *security association* is the set of all information system resources (i.e., security and communication protocols, security functions, security services, and mechanisms) employed to securely link two distinct security contexts, sc_i and sc_j, on different end systems, es_i and es_j, supporting the same information domain, d.

An effective security association, in essence, extends the protection provided by the participating end systems through the transfer system across the communications network. Critical information needed to establish a security association is exchanged between end systems using a Security Association Management Protocol (SAMP). The protocol information is contained in the SMIBs of the participating information domain and end systems supporting the domain.[49]

3.4.4 Information Sharing and Transfer

The DGSA imposes certain restrictions on information sharing and transfer in order to maintain the integrity of information domains resident on end systems. As stated previously, the simplest method of sharing information objects is to accept new users into an existing information domain, thus giving the new members use of previously inaccessible objects. The users are granted privileges for the information domain that extend to all information objects in the domain. An alternative method of sharing information objects is more complex and requires creating a new information domain, establishing a new domain security policy, accepting new users into the domain, and populating the new domain with the information objects that need to be shared.

As stated in Section 2.3.2, information objects can be transferred between information domains only in accordance with the domain security policies of each participating domain to include the specific import-export rules established by the concerned authorities. Constraint 3 requires that a user transferring information objects between domains must be a member of both the source and destination information domains and must possess the appropriate privileges (to include release authority). There is, however, an additional constraint employed by the DGSA with respect to inter-domain transfer of information objects.

Constraint 4: Inter-domain transfers of information objects can occur only if the source and destination information domains are resident on the same end system and if the transfer agent is a user in both domains.

---

49. A SMIB data structure called the Agreed Set of Security Rules provides domain label information as well as cryptographic keying information for the security association [5].

This constraint implies that inter-domain transfers cannot occur among distributed systems (i.e., end systems and relay systems). The rationale for including this restriction involves the definition of distributed security contexts as discussed in Section 3.4.3. While an individual end system uses appropriate security mechanisms to ensure separation of multiple security contexts, a distributed security context employs cryptographic mechanisms to ensure strict isolation of information in transfer between end systems. An essential requirement in establishing a distributed security context is the sharing of key material and other information supporting the use of cryptographic mechanisms. Sharing key information from different information domains is much more difficult and results in additional complexity in communication and security protocols. Thus, for implementation reasons, the DGSA restricts inter-domain information transfer across multiple end systems. There is also an issue of trust. Distributed trust is much more difficult to assure than is centralized trust. If the owners of the information in both domains are willing to host their domains on a single end system, the import-export policy of the common system can be used to control transfers between domains with higher assurance.

Given the inter-domain restriction described by Constraint 4, it is possible to provide a complementary view of information transfer restrictions between end systems.

> **Constraint 5:** Distributed security contexts can support information transfer only within a single information domain.

This constraint implies that any transfer of information objects between two end systems must occur within the context of a single information domain. The rationale for including this restriction is, in essence, the same as for Constraint 4.

The EDS Model can be used to graphically describe the constraints outlined above. Referring to Figure 4, consider the *x*-*z* plane or space of information domains and end systems. Constraint 4 limits information flow to the *x*-axis only; that is, given a particular end system *es*, information objects can be transferred between any information domains in the set of domains resident (or hosted) on *es* in accordance with domain security policies. Constraint 5 limits information flow to the *z*-axis only; that is, given information domain *d*, information objects from *d* can be transferred between end systems in the set of end systems within the bounds of appropriately established distributed security contexts.

3.4.5 Multi-domain Information Objects and Policies

To support mission-related activities, it may be necessary to use information objects in different information domains and combine those objects into composite information objects.

The DGSA recognizes this requirement, but places severe restrictions on how this composition is accomplished. Through the security management facilities on the end system, users can create the perception that a set of information objects O_1, from information domain d_1, and a set of information objects O_2, from information domain d_2, form a single composite information object. These composite objects are in reality virtual objects, and are referred to as *multi-domain objects*. Multi-domain objects can only be used to print, display, or transfer information between end systems from multiple information domains in a sequenced order.

Multi-domain information objects are virtual objects in the sense that a real composite object is never actually created on the end system. A set of opaque object references (one possible implementation) located within the end system information domain can be used to provide identification of those constituent objects required to form the composite information object. It is important to maintain the security concept of strict isolation during the process of using multi-domain objects and it is always the case that the constituent objects of the multi-domain object must be protected in accordance with the security policies of the information domains where the objects reside.

> **Constraint 6:** Each information domain providing a constituent information object for a composite multi-domain object must be supported on each end system participating in the transfer of that multi-domain object.

Constraint 6 is employed to facilitate a transfer of a multi-domain object between end systems. The requirement to support all information domains contributing to the formation of the composite information object ensures that a separate set of distributed security contexts can be established for each constituent component of the multi-domain object. That is, the security association between end systems must establish a separate set of security contexts for each component of the composite object deriving from a different information domain, thus maintaining strict isolation between domains while the information objects are in transit across the network. This is analogous to having multiple secure channels between the end systems. Figure 8 illustrates a conceptual view of a multi-domain object transfer between end systems.

3.4.6 Uniform Accreditation

The DGSA generates certain accreditation requirements for information domains. The fundamental security concepts described in the goal security architecture (i.e., information domains, strict isolation, absolute protection) provide the basis for achieving a *uniform accreditation process* for the *implementations* of information domains. In essence, each implementation of an information domain must be accredited to process information on its supporting end

system (and relay system) as part of an LSE and to transfer information over each LCS and CN supporting the distributed system. That is, given an enterprise with a set of end systems *ES* and a set of information domains *D*, each information domain $d \in D$ must be accredited to process information on a set of end systems $ES_d \subseteq ES$. A uniform accreditation process must ensure that each information domain security policy is enforced on each end system where processing will occur. The objective is to achieve consistency of protection on all end systems supporting a given information domain by providing at least the minimum strength of protection on each end system necessary to enforce the domain security policy.

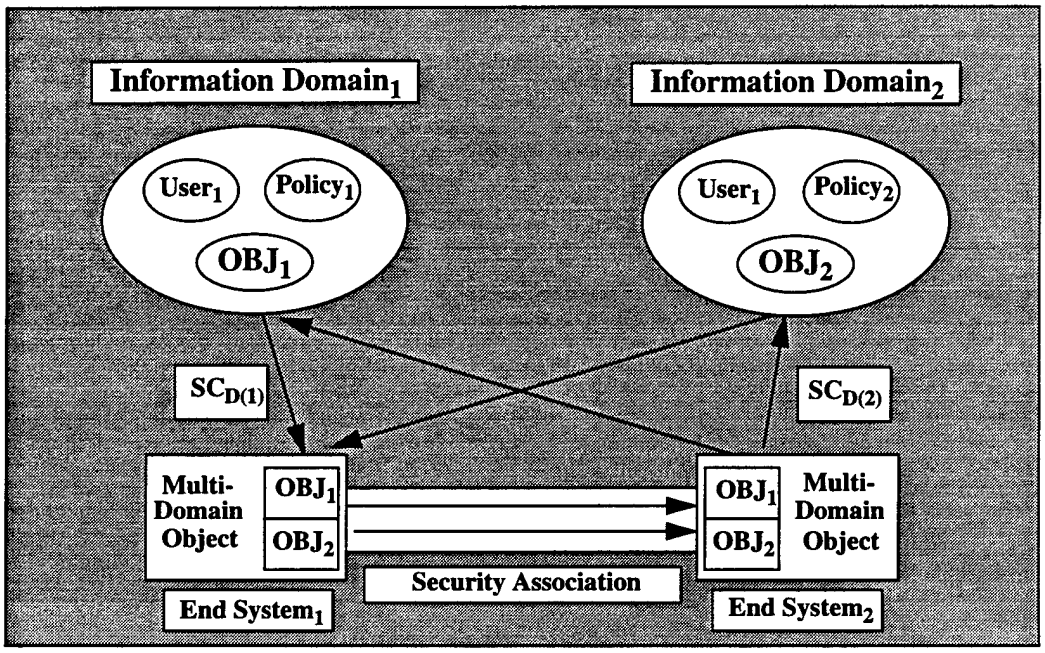

Figure 8. Multi-Domain Information Object Transfers

The concept of information domains provides a logical grouping of information, users, and a security policy for a community of interest that establishes a manageable boundary around a set of controlled entities. The concept of strict isolation provides the basis for information domain independence and separation on end systems. The concept of absolute protection provides the basis for ensuring that an information domain has the necessary and sufficient protections (as set forth in the domain security policy) within LSEs and across LSEs in a distributed systems environment. Employing all three concepts produces a synergistic effect that creates an environment conducive to achieving uniform accreditation.

From an implementation perspective, each information domain must have an accreditation authority responsible for obtaining the necessary evaluations of all LSEs and transfer systems supporting the accreditor's domain. The evaluation of each LSE and transfer system assesses the capability of each specific end system (or other LSE component) to support strict isolation of the information domain. The complete set of LSE evaluations assesses the capability to achieve absolute protection for the information domain. The results of the overall evaluation provide the information domain accreditor with a documented assessment of the level of residual risk assumed by placing the domain into operation.

4. Security Policy and DGSA Systems

This section discusses the relationship between the security policy for an information system and the resulting DGSA-style architecture. We first talk about this relationship in general. Next, an example security policy intended to exercise many of the features of the DGSA is defined. This is followed by the threats to the example system and the security services required to counter these threats. The policy is then expressed as a set of requirements. Finally, a DGSA-style architecture that satisfies these requirements is given, including informal proofs that the requirements are sufficient for the policy and that the architecture satisfies the requirements. Thus, any implementation of the architecture on a target system (such as TMach) will satisfy the policy.

4.1 The Effect of Policy on Software Architecture

The architecture of a system defines its structure at a high level as a set of components and the connections between them. It allows a system designer to concentrate on the ways in which the components interact without dealing with the details of the components themselves. For any system type (e.g., office buildings, highways, information systems) there are a small number of architecture styles, each with its own strengths and weaknesses, from which to choose. Thus, the designer of a new system starts with an architecture style, perhaps one selected from a taxonomy of architectures [7] to best meet the needs of the system or one that was used successfully in a similar system.

An information system is designed using several complementary architectures, each addressing a different concern. One architecture describes the hardware components and how these are connected to form distributed or networked systems. A second architecture, the *software architecture* [7], describes the computational components and how these are interconnected; styles include pipes and filters, object based, and layered. The hardware and the software architectures are somewhat independent in that many software components may be placed on a hardware component, or a single software component may be distributed across many hardware components (although performance and fault-tolerance considerations normally strongly

influence the mapping between the architectures). A third architecture, the *information architecture*, deals with components of information and the ways in which they are related.

The DGSA defines an information architecture style (or generic architecture) in which information is organized in components (information domains) on the basis of the required protection that must be applied to the information. The connections between components are formed by the ability to transfer information from one domain to another. The DGSA components are independent of components of the hardware and the software architectures. For example, information domains can cross client-server software architecture boundaries by grouping objects managed by different servers in a single information domain, and by placing objects managed by a single server into multiple information domains. Likewise, the objects of an information domain can be distributed across several hardware platforms.

The system security policy, by restricting how the various information objects are managed, accessed, and distributed, effectively partitions objects among the information domains and also defines the information domain security policies. For a DGSA system, all objects in an information domain are managed, accessed, and distributed in the same way. If a particular user is permitted to observe one information object but not another, those objects are accessed differently and must be in different information domains. A system security policy differentiates among objects on the basis of security attributes. For example, permission to access a file might depend in part on the file attributes *owner*, *owner permission vector*, and *other permission vector*. If the policy requires that any access to a Top-Secret object be audited, the object attribute *level* is used to determine which accesses to audit. Thus, as specified in Constraint 1 of Section 2.3.1, the objects in an information domain will all have the same values for each of the security attributes.

An information domain security policy is determined by limiting the system security policy to the objects in the domain, replacing references to the object security attributes by the values fixed for the domain. Consider a MultiLevel Secure (MLS)[50] system security policy that specifies that a user only be able to modify those objects for which the object's *level* dominates the user's *level*. An information domain for which the *level* value is *secret* will have a policy requirement: "a user may modify an object of the domain only if *secret* dominates the user's *level*". This requirement is further specialized by replacing the term *modify* with the operations

50. An MLS policy classifies data according to a set of sensitivity levels: Unclassified, Confidential, Secret, and Top Secret. Users are also assigned to one of these levels as a measure of trust. These levels are related by a relation *dominates* such that Top Secret dominates all of the other levels, Secret dominates Confidential and Unclassified, and Confidential dominates Unclassified. These levels may be extended with categories, and there also may be "need to know" requirements.

of the domain (such as *write* and *append*) that change the value of objects: "a user may write or append an object of the domain only if *secret* dominates the user's *level*".

An information system has many requirements, other than those derived from the security policy, that also affect the software architecture. The sets of information objects and users, and the connections between them, are primarily determined by the processing required to accomplish the mission for which the system is used. Also, some security requirements, such as avoiding denial-of-service attacks, might affect system functions such as scheduling that support the software architecture, rather than affect the architecture itself.

A DGSA-style software architecture is generally independent of the hardware configuration. It can be run on a single end system, such as TMach, or distributed over several end systems, such as Triad [22]. Of course, features such as replication for fault tolerance are better suited for a distributed environment where the replicas are executed on processors with different failure modes, and the distribution of software architecture components over end systems may affect the efficiency. Also, each system component must provide the security services required by the software architecture components that it supports.

4.2 Application-Level Statement of Example Policy

The security policy for an information system depends on the mission for which the system is to be used and the laws and regulations governing that mission. The security services provided to enforce that policy depend on the threats to the system. In this section, we first present the mission for our example system. We then give a generalized security policy for this mission; this policy could be specialized by requirements generated from regulations governing a particular situation.

The system mission was chosen so that its security policy exercises as many of the features of the DGSA as possible, in order that support for the DGSA by a target operating system can be properly evaluated. These features are as follows:

- A variety of security services, to include authentication, access control (including type enforcement [2]), data confidentiality, and data integrity.
- Security management functions, including auditing.
- Information transfer between domains.
- Contexts shared between multiple LSEs.
- Strict isolation

Another important criterion for the example system is that the mission be real and that it have applications in both military and commercial networks. A contrived example would leave doubts about the usefulness of the DGSA for interesting applications, while an example that is exclusively military would not test the generality of the DGSA.

4.2.1 Example System

The system considered here is a guard, the mission of which is to control the flow of information from one environment to another. Such a guard might be used between military Local Subscriber Environments with different security levels, or between a corporation's internal network and the outside world. The controls included are *filtering* and *sanitizing*. Filtering performs checks on the data, such as a 'dirty word' search, a format check, or a check of sender and receiver addresses, and prevents any information failing a check from reaching the destination environment. Sanitization removes some of the detail, such as precision in a number or detail in an image. A particular guard might delete the names of towns under consideration for a new factory and block binary data or messages broadcast by political candidates.

The particular filter and sanitization functions are left unspecified, so that the guard is adaptable to the needs of a variety of situations. The type of filtering and sanitizing depends on the information that passes through the guard, and the strength of the filtering and sanitizing needed depends on the sensitivity of that information and the level to which the source and destination environments are trusted. Also, the guard must be able to mature over time, with the set of these functions available changing in response to changes in filtering technology and to threats to the system. To satisfy these requirements, each type of filtering is defined as an operation called a *filter* and each type of sanitization is defined as an operation called a *sanitizer.*

The guard processes information in units of *blocks*. The exact definition of a block varies with the situation for which the guard is used and is left unspecified here. A filtering function either accepts or rejects entire blocks. The formats of the blocks, referred to as the block *types*, depend on the situation. A guard may need to handle several block types, and the filters and sanitizers applied to a block depend on its type. For example, a guard might need to handle block types **email** and **spreadsheet**, filtering any email block sent to a competitor and sanitizing each spreadsheet block to remove fractional portions of numbers.

Other applications may run on the end systems that support the guard. However, normally the source of the information and its destination will be on other end systems, one of which is in a different LSE (the guard can be used to screen information either coming into or leaving an LSE).

4.2.2 Policy

A high-level statement of a possible security policy for the guard is presented here. It will be refined into a list of requirements in Section 4.4. Since automatic filtering is imperfect (in general, it is an unsolvable problem), we provide for spot checks by a human. We also assume an audit mechanism that can be used to detect attacks against the system.

> Any information originating in the source environment must be sanitized and filtered as specified by the System Security Officer (SSO) for the type of the information, or must be approved by a Person in the Loop (PiL), before entering the destination environment. The filtering and sanitizing functions must be independent of any other filtering. Blocks are randomly sent to the PiL for checking instead of being filtered, with the percentage of the blocks of each type specified by the SSO[51]. Attempts to ship improper information through the guard shall be audited.

In the terminology of integrity policies [3], filtering is provided by Integrity Verification Procedures and sanitization by Transformation Procedures. For each block type, these form a trusted pipeline [2], from the source environment to the destination, through which blocks of that type pass and that cannot be bypassed. The independence of the filters and sanitizers ensures that the only information flowing through the pipelines is that contained in the blocks. Thus, experimental or untrusted filtering or sanitizing code can be used with assurance that any outputs from them will pass through a final trusted filter before reaching the destination.

4.3 Threats and Services

The security services required to enforce the policy, the allocation of these services to system components, and the strength of these services are determined based on the perceived threats to the system. For the example mission and policy, the threats and services are listed in Table 1. These services are allocated to the LSEs on which the guard executes.

---

51. The PiL could also check blocks that have been filtered, but that complication will not be addressed here.

Table 1. Threats and Countermeasures for Guard

Threat	Description	Harm	Security Service
Masquerading	An unauthorized person claims to be the SSO, PiL, or auditor.	Unauthorized Disclosure and Manipulation, Denial of Service	Authentication
Improper Access	A block bypasses filters and sanitizers.	Unauthorized Disclosure	Access Control, Code Integrity, Data Confidentiality
Insider Agent	Filters or sanitizers are replaced by a user other than the SSO.	Unauthorized Disclosure, Unauthorized Manipulation	Access Control
Collusion	Users cooperate to use weak filters for a block.	Unauthorized Disclosure	Access Control (separation of duty)
Malicious Software	Information that passes the filters is rejected or infinitely delayed.	Denial of Service	Code Integrity
Untrusted Software	Information from rejected blocks is inserted into accepted blocks.	Unauthorized Disclosure	Access Control Data Integrity

The required strength of the authentication service depends on the criticality of the information passing through the guard and on the ability for users other than a SSO, a PiL, or an auditor to gain access to the system. An unauthorized user acting as the SSO could change the set of filters so that improper data blocks are approved and good blocks are rejected, or change the set of sanitizers so that the blocks are modified. An unauthorized user acting as the PiL would be able to observe information that has not yet passed through the guard and also approve improper data blocks. An unauthorized user acting as the auditor could observe data that has been rejected by the guard or the PiL.

Strong code integrity is required for any trusted code, including the code that routes blocks through the guard and the code that determines if a block has passed the necessary filters. Incorrect code could cause a wrong set of filters and sanitizers to be used, for information that has been rejected by the guards to be passed on, or for approved information not to be passed on. The code actually executed must be the intended code and it must be correct. Weak code integrity is satisfactory for the filters (additional assurance can be gained by using extra filters that duplicate some of the checks), allowing experimental untrusted filters to be used.

A strong access control service is required to prevent unauthorized users from accessing information contained in the guard. Security depends on knowing what user is responsible for

each access and then preventing unauthorized accesses. Additionally, strong access control is required to constrain low-integrity code, such as experimental filters. Filters must not be allowed to modify blocks, and filters and sanitizers must not be able to modify decision tables used by the guard or to store information from one block for use when processing a later block.

The required strength of confidentiality for any data being processed by the guard depends on the criticality of that information. Moderate confidentiality is required for the code that is executed and any decision tables (analysis may reveal weaknesses that can be exploited).

The security policy could be violated if the PiL approves a block containing information that should not be released, or if the SSO specifies the wrong set of filters and sanitizers for some information type. Therefore, the PiL and the SSO are trusted to behave properly. This trust is supported by the requirements for authentication for these roles and for code integrity for the code they execute, as described above.

As described above, the guard itself provides a data integrity service to the rest of the system. A filter only certifies blocks that pass its integrity checks. Also, a sanitizer could be used to encrypt blocks to provide data confidentiality.

4.4 System Requirements Derived From Example Policy

From the high-level example policy, the following list of requirements can be derived. We then sketch a proof that these requirements are sufficient to ensure the policy. (They may not all be necessary, however. Choosing from all appropriate lists is left to the system architect) The proof also serves to show the connection between the policy and the requirements; the requirements were refined while developing the proof. In Section 4.5 the requirements will be used to help define the set of information domains and then will be allocated to the domains to which they apply.

Requirements on users:

1. It is possible to partition the users of the system, except the auditor, into two groups. All users that share information with the guard's source must be in the first group and all users that share information with the guard's destination must be in the second group. Users that do not share information with the source or the destination can be placed in either group, consistent with requirement 11.
2. The auditor shall not release information to the second group or to the destination.
3. The users acting in the SSO, PiL, and audit roles must be different from each other and from the users that determine the type of each block.

4. Any user acting in the SSO, PiL, or audit role must be identified and authenticated.

Requirements on the System Security Officer:

5. The SSO shall specify a list of block types.
6. For each block type, the SSO shall specify a list of filters and sanitizers.
7. For each block type, the SSO shall specify the percentage of the blocks that are randomly sent to the PiL instead of passing through the filters and sanitizers.
8. Changes to the filter and sanitizer lists or to the percentage of blocks sent to the PiL shall be audited.
9. For each block type, a record of the number of blocks of that type that have been accepted and the number that have been rejected is kept. The SSO can request that these records be sent to the auditor, at which time the records being kept are reset.

Requirements on the blocks:

10. Each block passing through the guard shall be associated with its type.
11. Only blocks that have been accepted by all filters and sanitizers specified for their associated type, or by the PiL, can be accessible to users from both groups.
12. Any blocks that are rejected by either a filter or the PiL and the reason for rejection shall be audited.
13. The actions of a filter or a sanitizer on a block must be a function only of the value of that block.
14. Only the sanitizers and the PiL may change the contents of a block.

Correctness requirements:

15. A filter shall reject any block for which the checks performed by the filter on the block fail.
16. Any code that can change the security attributes of some data must be correct to a high degree of assurance.

These requirements can be seen to satisfy the policy as follows:

- Any information originating in the source environment must be sanitized and filtered as specified by the SSO for the type of the information, or must be approved by a PiL, before entering the destination environment —

  From requirement 1, information can only pass from the source environment to the destination environment by being in an information object accessible to both groups of users, or by way of the auditor.
  From requirement 2, information cannot flow from the auditor to the destination environment.
  From requirement 11, only blocks that have been accepted by all filters and sanitizers specified for their associated type, or by the PiL, can be accessible to users from both groups.

  Thus, any blocks that pass from the source environment to the destination environment have been accepted by all filters and sanitizers specified for their associated type, or by the PiL.

  From requirement 10, each block is associated with its type.
  From requirement 6, the SSO specifies a set of filters and sanitizers for the type.
  From requirement 15, a filter accepts only those blocks that pass the checks it performs.
  From requirements 13 and 14, information from a rejected block cannot be inserted into an accepted block.

- The filtering and sanitizing functions must be independent of any other filtering — From requirement 14, filters do not change the value of a block, and from requirement 13 the actions of filters and sanitizers depend only on the value of the block.
- The PiL randomly checks blocks, with the percentage of the blocks of each type specified by the SSO — implied by requirements 5 and 7.
- Attempts to ship improper information through the guard shall be audited — implied by requirements 8, 9, and 12.

There are several additional requirements that are not needed for the policy, but instead are introduced to counter threats listed in Section 4.3:

- Collusion — requirement 3
- Masquerading — requirement 4

- Malicious Software — requirement 16

4.5 Information Domains for the Example Policy

The requirements presented in Section 4.4 will now be used to define a set of information domains, each containing a set of information objects and a set of users and having a security policy. This security architecture, showing the information domains as boxes and the connections by which information objects can move between domains as arrows, is given in Figure 9. These domains specify the different ways in which information is protected at differ-

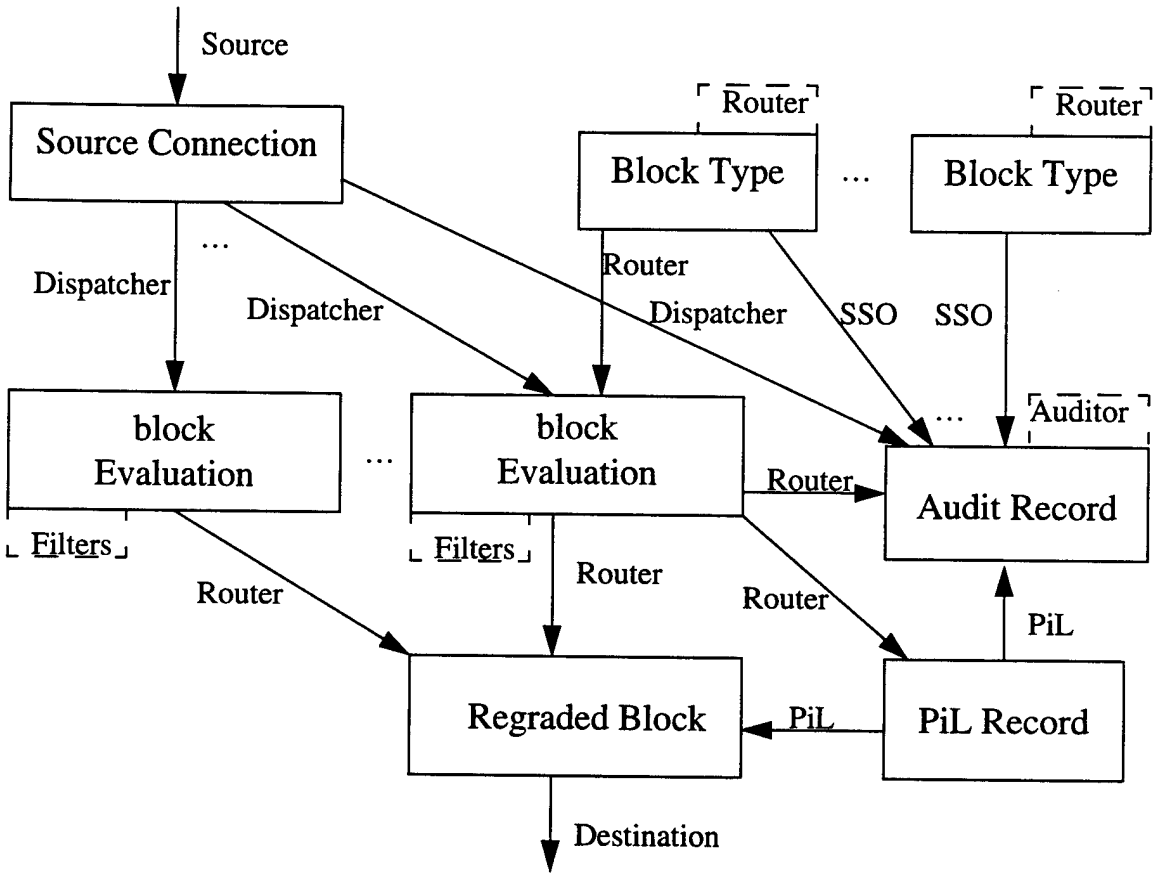

Figure 9. Information Domains and Users for Example Policy

ent points in the pipeline; they are independent of the end systems on which the information may be located, or the software architecture used to represent and process information objects. Thus, a block passing through the guard may remain on the same end system and be managed by the same object manager while flowing from the Source Connection domain to a block Evaluation domain to the Regraded Block domain. Conversely, a block might move from one end

system to another and be passed from one manager to another while remaining in the Source Connection domain. A description of each domain will be given in subsequent sections, specifying the different forms of objects in the domain, the users that are members of the domain, and the domain security policy.

Several new 'users' are introduced to facilitate transferring information objects between domains and to carry out the functions of the guard. These could have been consolidated into a single internal user, but we chose instead to assign each function to a different user. The users that transfer objects between domains are a Dispatcher and a Router for each block type. Each filter and each sanitizer is represented by a user, which allows the Router to control the sequence in which these operations are performed by controlling the order in which the filter and sanitizer users are admitted as members of a block Evaluation domain.

Requirements 2, 5, 11, 13, and 14 restrict how information objects can be assigned to domains. Requirements 1, 5, 6, 7, 9, and 11 help define policies for various information domains. Requirements 8, 10, and 12 specify conditions for setting up multidomain objects. Requirements 3 and 4 deal with authentication, which is a service that supports the architecture rather than an architectural component. Requirements 15 and 16 deal with properties of code rather than data.

The partitioning required by requirement 1 places the Source user and all users that operate the guard (the Dispatcher, the SSO, the PiL, Routers for each block type, and the Sanitizers and Filters) into the first partition and the Destination user into the second partition.

4.5.1 Source Connection

This domain is the entry point to the guard. A Source user deposits a block here for processing. The Dispatcher determines its type; if the type cannot be determined, the block is rejected and sent to the Auditor. Otherwise, the Dispatcher creates a new block Evaluation domain for filtering and sanitizing the block and exports the block to it.

Objects: source blocks, blocks with rejection notice

Users: Source, Dispatcher

Policy: Source may: import blocks from other domains

Dispatcher may: read (to determine type),
create a block Evaluation domain with users
Dispatcher and Router (for the block type),

export to block Evaluation (information),
append reject notice, export to Audit Record

4.5.2 Block Type t

For each block type **t**, there is a Block Type information domain and a user Router. Each type has a list of filters and sanitizers that only the SSO is allowed to modify, thus satisfying requirement 6. It also has a probability, set by the SSO, that a block of that type should be sent to the PiL rather than filtered, thus satisfying requirement 7. The action of modifying the filter/sanitizer list or setting the PiL probability has a side effect of notifying the Auditor, satisfying requirement 8. The router keeps a record of the numbers of blocks that have been accepted and rejected, that the SSO copies to the Audit Record and resets, thus satisfying requirement 9.

Objects: filter/sanitizer list, accept/reject record, PiL probability

Users: Router t, SSO

Policy: SSO may: modify filter/sanitizer list
specify PiL probability
copy filter/sanitizer list or PiL probability to Audit Record
copy accept/reject record to Audit Record and reset

Router t may: observe filter/sanitizer list, observe PiL probability,
copy filter/sanitizer list to block Evaluation,
increment accept/reject record

4.5.3 block Evaluation

There is one of these domains for each block being filtered and sanitized. It is used to provide a unique security context for the block, thus satisfying requirement 13. The Dispatcher creates a domain when it receives a block from the source, with the only user initially being the Router for the type of that block; this Router creates an association between the type **t** and the block, satisfying requirement 10. The Router alters the user list to contain the next Filter or Sanitizer. The sequence of allowed Filter and Sanitizer users forms the secure pipeline. The Router exports the block to the Auditor Record domain if a Filter appends a reject notice (also incrementing the reject record in the Block Type domain), partially satisfying requirement 12, or to the Regraded Block domain if all necessary Filters and Sanitizers have signed the block (also incrementing the accept record in the Block Type domain). Only Sanitizers may modify a block, satisfying requirement 14.

Objects: signed block, block with rejection notice, filter/sanitizer list

Users: Router t, Filters, Sanitizers, Dispatcher

Policy:

Dispatcher may:	import from Source Connection
Router t may:	import from Block Type t, modify user list of policy, observe signature list, observe filter/sanitizer list, export to PiL Record, export to Regraded Block, export to Audit Record
Filter may:	observe, sign, append reject notice
Sanitizer may:	observe, modify, sign

4.5.4 PiL Record

The PiL shall append a reject notice for any block that it rejects to the rejected block and export that block to the Auditor, partially satisfying requirement 12. It exports any blocks that it accepts to the Regraded Block domain.

Objects: blocks, blocks with rejection notice

Users: PiL, Router

Policy:

Router may:	import from Internal Block
PiL may:	export to Regraded Block (accepted blocks), append reject notice, export to Audit Record

4.5.5 Regraded Block

This domain contains the blocks that have been accepted by all of the applicable filters and sanitizers, or by the PiL, and is the only information domain accessible by users in both partitions, satisfying requirement 11.

Objects: blocks

Users: Router, PiL, Destination

Policy:

Router may:	import from block Evaluation
PiL may:	import from PiL Record
Destination may:	export

4.5.6 Audit Record

This domain is used to satisfy requirement 2. Various users may import audit records, but only the Auditor may observe these records once in the domain and the Auditor may not export these records. (The operations performed by the auditor are outside the scope of this work and therefore additional objects may be required. Also, we do not specify a policy for the auditor.)

Objects: audit records, ...

Users: Auditor, Dispatcher, Router, PiL, SSO

Policy:

Dispatcher may:	import from Source Connection (bad type)
SSO may:	import from Block Type (filter lists, records)
Router may:	import from block Evaluation
PiL may:	import from PiL Record (rejected block)
Auditor may:	...

4.6 Security Management Information Base

All security-relevant information pertaining to an information domain is contained in a Security Management Information Base (SMIB). A SMIB is either an information domain used in the management of one or more information domains (one-to-one or one-to-many), or a set of information objects that are part of the information domain being managed (embedded). For each information domain, information objects that might be part of a SMIB include:

- Security policy rules
- Object security attribute values for an information domain
- Values for the security attributes of each member user
- Member authentication information
- Visible security label information (for use when displaying the information)

Operations on the SMIB include creating and destroying information domains and changing the set of member users of an information domain. Other operations might return object or user security attribute values.

The security policy for the SMIB defines the criteria for security administration. This policy defines which users may access the SMIB and what operations they may perform. Thus, it defines who may create information domains and who can change the set of users in a

domain. Of particular importance are the restrictions on modifying the security policy rules, which control how the policy can be adapted for changes in the mission or situation.

The SMIB for the guard information domains described in Section 4.5 would be in several additional information domains. The first of these domains contains information objects used to manage information domains that are created when the guard is started, including the Source Connection, the PiL Record, and the Regraded Block domains. Other SMIB domains contain information objects for the block Evaluation domains. The information objects for the Block Type domains might be in the first SMIB domain if the types are set when the guard is created, or in an additional domain if they may be created and destroyed.

4.6.1 Security Policy Decision Function

An important operation on the SMIB is the Security Policy Decision Function (SPDF), which is used whenever an operation is attempted on a managed information domain. The SPDF provides the security policy rules for the operation, including any requirements for authentication and auditing and whether the attempt is allowed. Its parameters may include the information domain of the object being operated on, the user that requested the operation, and the operation. If the security policy for the information domain is history based, the SPDF may modify the SMIB objects for that domain.

The security policy for the SMIB specifies which users may execute the SPDF. Normally, this permission is given to those users that must enforce the security policy for information domains, such as a type manager that implements the operations for a particular type of operation. Restricting use of this function is especially important when the security policy is history based, in which case use of this function may affect future decisions.

4.6.2 Block Evaluation Domain SMIBs

There is a SMIB information domain for each type of block, with the Dispatcher, the Router for that type, and the manager that enforces security for that type as users. Each block Evaluation domain is represented using an information object in the SMIB information domain corresponding to the block type **t**. This information object contains a list of operation sets indexed by the users that are members of the represented domain. As described in Section 4.5.1, a block Evaluation information domain is created by the Dispatcher, and as described in Section 4.5.3, the user set is modified by Router **t**.

As part of the action of creating a block Evaluation information domain, the Dispatcher creates an information object in the SMIB domain corresponding to the block type **t** of the

block Evaluation domain. This information object contains the initial security policy for the new information domain, with the operation lists for the Dispatcher and the Router **t** users as given in Section 4.5.3.

The Router changes the user set by creating an information object in the block Evaluation domain that contains the new set of filters and sanitizers that should be enabled for that domain. It exports this object to the SMIB domain and uses this set to replace the current filters and sanitizers in the information object supporting the block Evaluation domain.

The SMIB information domains in support of block Evaluation domains can be described as:

Objects: operation lists indexed by user, filter/sanitizer sets

Users: Dispatcher, Router **t**, Security Enforcer

Policy:

Dispatcher may:	create operation list object
Router **t** may:	import from block Evaluation, replace filters/sanitizer users in operation list, delete
Enforcer may:	invoke SPDF to determine the security policy decision

5. Services in Support of Security Management

Security management [10] is concerned with the services needed to monitor, control, and coordinate the entities that form an information system. The entities in a DGSA-style system are users, information objects, and resources. Information domains define the relationships between the entities, and security associations the relationships between domains on different LSEs. The services needed to manage these entities and relationships are in addition to those described in Section 4.3 in support of the security policy for the guard application.

5.1 Support Supporting Users

Security within a DGSA-style system depends on identifying the user responsible for each access to an information object within an information domain. The responsible user must be a member of the domain, and security attributes of the user are used to make security policy decisions. Thus, each process that is capable of requesting accesses on information objects must be identified with a user (those system processes that provide services in support of the architecture do not need this identification). This identification must be available to the Security Policy Decision Function (SPDF) at the point that an access is attempted.

The operation of identifying a process with a user must be controlled. Depending on the security policies supported by a system, a user might represent an individual, a system function, a role, a group of individuals, a particular individual acting in a role, several users acting together, one user speaking on behalf of another, or one of these users operating with a restricted set of rights [1]. Identifying a process with an individual might require either action by another process identified with the individual, authentication through a device, or possession of a cryptographic key. Identifying a process as a system function might be allowed during system initialization or as the result of action by a process identified with the SSO. Identifying a process with a role might require action by a process identified by a user authorized for that role. Identifying a process with several users might require separate actions by each of those users.

Processes can be managed as information objects that belong to domains. Operations on processes include creation with a particular user identity, destruction, changing the user

identity, signing a process, or restricting a process' own user identity. Other operations control the allocation of system resources to the process, including processing time and memory. The security policy of the process domains restrict how user identities are assigned.

A security context supports a single user operating in an information domain. Different security contexts may only interact through shared information objects in accordance with the security policy. Thus, the address spaces of security contexts that share an information domain may both contain the programs that implement the operations of that domain and the objects of that domain. However, the information system resources allocated to those contexts must be disjoint. In particular, there must be a different process state for each user of an information domain. Likewise, each user must have a different process state for each information domain in which that user operates.

5.2 Representation of Information Objects

A LSE must be able to control access to each of its individual information objects, as specified by the SPDF for the information domain that contains the object. Since for any particular user the allowed accesses must be the same for all the objects within that information domain, the domain could be handled by the LSE as a single information collection. However, services must be available for dynamically creating and destroying objects within the domain. Also, the objects may be of different types for which the accesses may be defined differently, and some accesses may not be defined for certain types. For example, the Information Block Evaluation domains contain both signed information blocks, and filter lists for which signing has no meaning.

One method for implementing information objects is within type managers implementing accesses to the objects. These type managers must record the identity of the information domain with the object and enforce the security policy of that domain. When an access is requested, the type manager invokes the SPDF for the domain, with the identity of the user requesting the access and the type of access, and receives the policy for that user, information domain, and access. Note that with this implementation there is a Security Management Information Base (SMIB) type manager that implements the SPDF. The strength of the security provided depends on the reliability of the type and the SMIB managers.

The security services provided by many LSEs consist only of identification and authentication for processes and access control lists for file system entities. For these systems, the previous solution can be simulated by creating a user for each object type and a file for each object. Only the owner of an object — the user corresponding to the type of the object — is allowed

to access that file. A set of processes belonging to the type user form the type manager. Other users access an object by sending a request through a pipe from which only the type user can receive. Modification of the file system directories must be restricted to type users; otherwise arbitrary users could create files that are then used as channels between DGSA security contexts, thus violating the principle of strict isolation. All accesses to directories containing the files representing information objects should be restricted to type users; otherwise users can exchange information by creating or deleting objects and observing the number of files in a directory. The strength of security depends on the reliability of the type and SMIB managers, and also on the strength of the file access control mechanism.

With this last implementation technique, an access control security policy for a **file** type with accesses read, write, and execute can be implemented directly without the use of a manager. Only the SMIB's users are allowed to set and modify the access control lists for these objects, or to create objects of type **file**.

5.3 Information Domains

An LSE must provide services that allow management operations on information domains, including creating, destroying, and modifying them. An information domain is represented on the LSE as several objects in the SMIB. To create a domain without any objects, the creator specifies the security policy rules (this may be done by identifying a SPDF that implements those rules), values for the object security attributes, an initial set of users, and security label information, as described for the Information Block Evaluation domain in Section 4.5.3. Changing the set of users of an information domain is also done by changing the SMIB. Thus, managing an information domain is controlled by the security policy for the SMIB.

Information domains can be modified by transferring (moving or copying) information objects between domains. Copying can be done using a service that creates a new object and copies the value of the information being imported into it. Moving an object from one information domain to another can make use of the copying service and also one that deletes the object from the old domain, but this can be inefficient especially if the object is large. For example, in the guard an information block object might be moved into the Source Connection domain, then to an Information Block Evaluation domain, then to the Regraded Block domain, and finally to a domain for further processing, thus requiring that it be copied four times. An alternate service is one that changes the domain identifier of the object.

Control of these transfer services is subject to the security policies for the domains involved and is therefore specified in the SMIB. The exporting domain must include permission for either copying or moving the object, and the importing domain must include permission for the same user to import objects.

5.4 Security Associations

Objects maintained on different LSEs must sometimes be organized into a single information domain. The Source Connection domain for the guard, described in Section 4.5.1, might include information blocks produced by the Source user on one LSE and also those being submitted to the guard on an LSE devoted to that guard. Likewise, the PiL might operate on its own LSE, requiring the Information Block Evaluation domain that supplies inputs to the PiL and the Regraded Block domain to which the PiL outputs are directed be on both the guard and PiL LSEs. A security association between these LSEs insures that the domain security policy is enforced on all of the LSEs supporting an information domain and also on communications between them.

A security association can only be established across LSEs that trust each other to enforce the security policy for that domain. As part of establishing this trust, the LSEs should be certified to supply the required services at the appropriate strength. There must also be agreement among the LSEs as to the set of users that are members of the domain, and thus what users may modify or observe the information in the domain.

When an information object is transferred from one LSE to another, it must be placed in the same information domain from which it was transferred. Thus, the information domain must have a common name on which both LSEs agree and that is used to label the object during transit. If this common name is different from the local name that an LSE uses for the domain, the LSE must provide a name service that makes the translation. Likewise, the name service may have to mediate the names for the users, the information objects, and the accesses of the domain.

In general, the objects of an information domain will not exist at some instant of time on all of the LSEs participating in a security association. When a user operating in a domain on an LSE makes an access to an object, that object potentially will reside on another LSE and must be located. Thus, the LSEs participating in a security association must have a location service (perhaps combined with the naming service). The service must then decide whether to move the requested access to the LSE of the object or to move the object to the LSE of the access. In case there are multiple replicas of the object (for example, to provide fault tolerance

or increased efficiency), the location service must maintain coherence among the various replicas [18].

The transfer system used to connect the various LSEs involved in a security association may also include other LSEs. In order to preserve strict isolation, the transfer system must prevent these other LSEs from having access to the domain. The transfer system may therefore have to provide encryption such that information from the domain and its management cannot be observed or modified. It also must provide identification sufficient to prevent these other LSEs from masquerading as members of the security association.

The security policy for an information domain must be consistent across all LSEs participating in a security association for that domain. Since the policy is governed by information contained in the SMIB, there must be a security association for each of the SMIB domains, that distributes it across the appropriate LSEs. The coherence protocol for this policy information should match that of the information objects that are subject to it.

6. Trusted Mach (TMach)

6.1 Selection Criteria

The goal of this project is to determine what would be required to implement DGSA on a commercially available operating system. To help make this determination, we needed to design an implementation for one such system. We chose Trusted Mach (TMach) [23] based on the factors listed below:

- Relevance to client. The National Security Agency had invested substantially in the TMach operating system.
- Commitment to commercialization. Trusted Information Systems (TIS) had decided to commercialize the system and had scheduled a version for commercial release in the first quarter of 1997.
- Strength of the operating system. TMach was undergoing evaluation at the B3 level under the U.S. DoD Trusted Computer System Evaluation Criteria (TCSEC) and B3-E5 level under the European Information Technology Security Evaluation Criteria (ITSEC) — both levels supporting mandatory access control via labels and auditing of important events.
- Suitability for commercial use. It appeared likely that a Microsoft Windows personality could be built on top of TMach — a necessity if the resultant was to be usable in the commercial world.
- Availability of implementation details and philosophy. Two of the authors of this report were able to participate in the architecture review with the designers and implementors of the system. TIS was willing to devote resources to providing answers to questions about TMach, its design and implementation.

6.2 Organization of the TMach System

Mach, from which TMach was derived, is a client-server (also called data abstraction [7]) architecture. Objects and their primitive operations are encapsulated in a server. An object

is "named" by a message *port* through which requests for operations are sent. The server for the object is the port's receiver.

6.2.1 Layers of TMach

In addition to being a Client-Server architecture, TMach may be viewed as a layered system, as shown in Figure 10. At the bottom is the hardware level containing an Intel X86 platform and a limited set of Input-Output devices. The lowest layer of software is the Open Software Foundation MK++ Mach Kernel [19]. The next layer contains a set of trusted servers that complete the TMach Trusted Computing Base (TCB). After that are sets of untrusted servers that define operating system "personalities"; while only a Posix personality has been built, other personalities such as Windows could be added. The top layer is a set of applications. An object layering and functional model is found in Figure 11. For greater detail, the interested reader may refer directly to the public documents on TMach available from TIS[52].

52.http://www.tis.com/docs/research/tmach/index.html

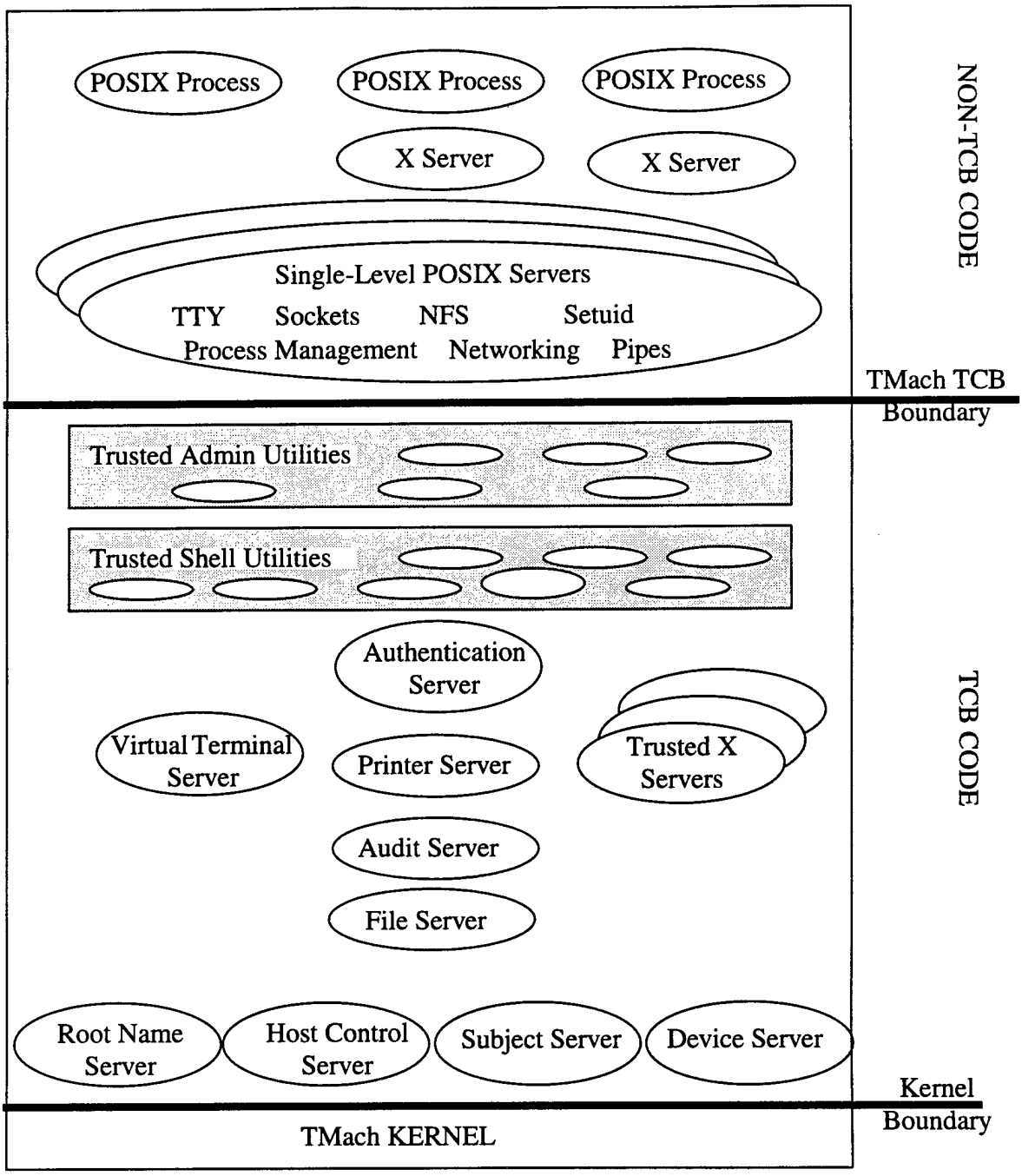

Figure 10. Trusted Mach Server Architecture

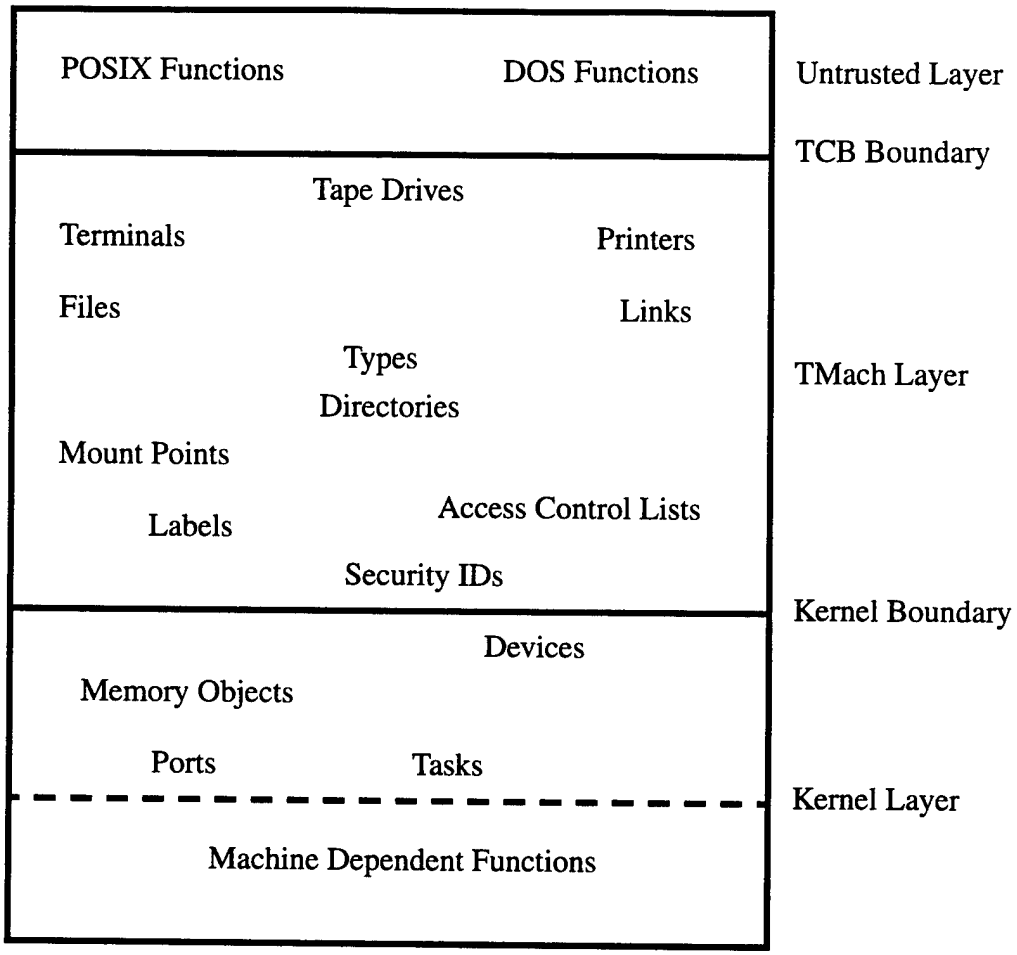

Figure 11. TMach Object and Function Layering

6.2.1.1 Hardware and Machine Dependent Layer

At the heart of a Mach system is a kernel that implements ports and other hardware-dependent features. The rest of a Mach system is implemented as a set of servers that can run on any hardware platform. Only the kernel runs in supervisor mode and has access to such hardware as device and page table registers. TMach uses the Open Software Foundation MK++ Mach Kernel.

The kernel provides an interprocess communication mechanism by which clients and servers can communicate. Communication takes place though *ports* — unidirectional commu-

nication channel, accessible only via send/receive capabilities called port rights. The information passed through ports is organized as *messages*, which may also contain send and receive rights.

6.2.1.1.1 Kernel-Supported Objects

The kernel is the server implementing several types of objects:

- Task - A task (process) is a collection of resources including a virtual address space consisting of a sequence of pages, and a port name space containing a sequence of port rights.in which one or more threads of execution operate. Each task has a Security Id which is used to tag any messages that the task sends. Servers and clients are implemented as tasks.
- Thread - A thread of control is one sequence of executing instructions within a task. All threads of a task share the resources of that task. in a given address space. Mach threads are considered lightweight (low overhead).

6.2.1.1.2 Security Constraints Enforced by MK++[53]

- The kernel applies the correct message tag to each IPC message sent from one task to another.
- The kernel applies the kernel's message tag to all messages it sends.
- The kernel permits a task to access a port only if the task holds a port right to that port.
- The kernel does not allow a task to map into its address space a memory object via a memory object representative that does not represent the identity of that task.
- The kernel does not allow access to a memory object in an access mode not allowed by the memory object representative through which access to the memory object was obtained.
- The kernel rejects any device operation via a device port that does not represent the identity of the requesting task
- The kernel rejects any device operation that implies an access mode not permitted by the device port through which the device is being accessed.

53.quoted from [19], page 65.

- The kernel's device drivers maintain the integrity of data on their associated storage devices.

6.2.1.2 TMach TCB Layer

The next higher level supports higher level programming abstractions such as:

- Directories — An abstraction consisting of organized data elements that permits association of hierarchically structured names with a set of attributes such as size, creation date, modification date, sensitivity level, owner, type of object, access rights, etc.
- Types — In TMach, this abstraction consists of a data element labeling an object instance which identifies the item manager for this object instance.
- Files — An abstraction consisting of data elements serving as persistent representation of text, program, or data which may be created, accessed, modified, and destroyed by an API to the type file.
- Memory Object — A memory object is a collection of memory cells sharing the same protection and logical attributes, e.g., the attribute "level", the attribute "type", the attribute "owner", or the attribute "where", with the attribute value being: in-memory, on-disk, in-transition. Mach has the ability to change these attributes while the process is running.
- Symbolic Links — An abstraction consisting of a data entry contained in a directory which permits an alternative symbolic path to an entity; an alias for an entity.
- Mount Points — An abstraction consisting of a data entry contained in a directory which permits a naming hierarchy to appear part of a larger hierarchy.
- Labels — An abstraction consisting of data associated with each persistent representation of information and with each transient representation of information which represents the sensitivity and security categories of that information.
- Access Control Lists (ACLs) — An abstraction consisting of a data in the form of a list which provides attributes whose values determine the rights of a user, a role, a group, or an organization (when properly authenticated) to use the object(s) with which the list is associated in a way associated with the kind of right, e.g. create, copy, access, regrade.

6.2.1.3 Application Layer

This level provides application level interfaces which were to have included:

- POSIX Interface - Provides an API offering services as specified by the POSIX interface standard.
- DOS Interface - Provides an API offering services as specified by the Microsoft (or IBM) Disk Operating System (DOS).

TIS provides a POSIX (OSF1) API, but does not provide a DOS API. The POSIX interface sends messages to services provided by the TCB layer which satisfies the request, possibly by sending messages to other services in the TCB layer or the Mach kernel as required.

6.2.2 Components of the TMach TCB

The components used in the TCB mediate all requests from any application to the microkernel. Each component is designed so that it could operate as a task in its own isolated address space communicating via ports. The components are layered based on their dependence on other components as in a partial ordering. In some cases, the volume or frequency of communication justifies the inclusion of several components in the same address space.

6.2.2.1 Layer 1

Layer 1 consists of the Root Name Server, the Host Control Server, the Subject Server, and the Device Server. In summary this layer performs primary mediation for all requests requiring system resources.

6.2.2.1.1 Root Name Server

The Root Name Server (RNS) has two principal functions: the mediation of requests containing symbolic names or symbolic links and the management of typed items including those items it manages directly: directories, symbolic links, and types. In the first capacity, the RNS determines what accesses an authenticated user is allowed for a named object, based on the Bell-LaPadula Multilevel security policy utilizing security labels and an Identity-Based Access Control security policy using the user id and groups of the user. If accesses are allowed, it gives the user send rights to a port through which the object can be accessed, and it gives the item manager for that object an *object descriptor* containing receive rights to the port, the identity of the user, and the set of permitted accesses.

In its second capacity, the RNS performs item management for types. Item managers in general system use must register with the RNS. Then they communicate with the RNS about item creation, deletion, and access. Each object instance they create is marked with their item type code; each instance they associate with a symbolic name is marked with their item type code in the RNS. Item managers using the TIS TMach class libraries guarantee that RNS access control decisions are enforced on incoming requests, and verify that the correct client is making the request and that the client has rights to the type of access it requested.

6.2.2.1.2 Host Control Server

The host control server mediates the configuration and operation of the host machine involving clock, processor set, and miscellaneous services. It maintains items for each clock and processor set as well as a single miscellaneous control item. ACLs on these items determine which servers have access to each item.

6.2.2.1.3 Subject Server

The subject server mediates the creation of tasks with a security id (used as a subject id) different from that of the parent task. It provides a security ID service and a subject creation service. It maintains an internal database that specifies the rights of subjects to create tasks representing subjects and which other subjects may be created. Creation of tasks with TCB ids is strictly limited. Security ids are immutable and cannot be destroyed or transformed after they are created.

6.2.2.1.4 Device Server

This server mediates the use all devices in the system. It acts as a registry for system devices and is the item manager for the raw device type used by the machine dependent kernel. For example, the File Server's file service might use the partition service which would use the raw disk service to make a new entry on the disk.

6.2.2.2 Layer 2

6.2.2.2.1 File Server

The file server provides access to the contents of files contained in the file system. It uses the Device Server to access "the raw disk device". It uses the memory map of the processor to map files into the address space of the task.

6.2.2.3 Layer 3

6.2.2.3.1 Audit Server

The audit server manages the collection of all audited event data, filtering it as prescribed by the system's security officer (SSO), buffering it, and storing it in files as required by the SSO.

6.2.2.4 Layer 4

Layer 4 consists of the Virtual Terminal Server, the Printer Server, the Trusted Window Servers, and the Core Crypto Server. In summary, it controls all usable accessible output except magnetic tape.

6.2.2.4.1 Virtual Terminal Server

The Virtual Terminal Server manages all I/O to terminals. In particular, it is responsible for multiplexing virtual terminals onto physical terminals in such a way that only one session at a single sensitivity level is using a terminal at a time. It also must switch to a trusted path (communication channel) when the Secure Attention Key (SAK) is depressed. The Virtual Terminal Server also makes certain that any contents of the terminal is erased when switching from one session to another and that any required sensitivity labels are displayed.

6.2.2.4.2 Printer Server

The printing server virtualizes physical printers in much the same way as the virtual terminal server virtualizes physical terminals. It makes certain that any contents of a printer is erased when switching from one session to another and that any required human readable sensitivity label is displayed.

6.2.2.4.3 Trusted Window Servers

In contrast to the VTS, the Trusted Window Servers may maintain multiple windows at different security levels, each labeled with the required human readable sensitivity label, on the output device simultaneously. It assures that data is not transferred inappropriately from a window of one level to a window at another level in inappropriate fashion. They also monitor the input for SAK and provide a trusted path to users running windows-based applications.

6.2.2.4.4 Core Crypto Server

The core crypto server provides a single level service for encryption and decryption. The SSO must set the key by command line. Each time the level is changed, all device state is destroyed.

6.2.2.5 Layer 5

6.2.2.5.1 Authentication Server

The authentication server is responsible for databases related to identification and authentication of users and provides them with subject credentials for use in the system including their group, and organizational affiliation and any role that a subject may take on. While direct access to the databases is permitted for read, only the authentication server may modify the data bases.

6.2.2.6 Layer 6

6.2.2.6.1 Trusted Shell Utilities

Trusted Shell Utilities are a collection of programs (instead of a service). These programs utilize the trusted path and must behave in a trusted manner. Such programs as login are among the trusted shell utilities.

6.2.2.7 Layer 7

6.2.2.7.1 Trusted Administration Utilities

Trusted Administration Utilities comprise a large collection of programs required for the administration of the system. Included is a shell program which helps administrators to do their jobs correctly and which may use some of the Trusted Shell Utilities.

6.3 TMach Use of Mechanisms Provided by Mach

6.3.1 Use of Mach tasks

Mach tasks are used by TMach to isolate various activities by containment in different address spaces and to strictly control interaction between the activities either through shared memory or by enforcing by value transmission of all arguments of requests to specific entry points within tasks.

6.3.2 Use of Mach threads

Threads in TMach are used to permit multiple subtasks to be run "simultaneously" in user and system programs. A separate thread may be spawned to perform I/O or any other activity which may be blocked, such as remote procedure call. Each thread in a TMach task has the same subject id and shares all resources assigned to that task.

6.3.3 Use of Mach ports

Ports and messages are *critical* elements of Mach protection. Since each task is in a separate address space, the only means of communication between tasks is by using the kernel interprocess communication (IPC) and shared memory features. The kernel uses IPC and shared memory to implement ports. "Messages are passed between tasks via ports, which are queues that hold messages. Tasks can obtain port rights, which enable tasks to enqueue or dequeue messages from ports."[54] These port rights are *send*, *send-once*, and *receive*. These rights may be passed to other tasks. User level subroutines and TCB level subroutines format messages and send them over ports to receiving tasks. Messages can be used to return results to the requesting tasks. Use of ports protects receiving processes from the threat of bypassing argument checks and prevents reception of messages from unauthorized tasks. Messages are marked by the system with the subject id of the subject who sent them, providing the ability to do authentication in DAC process. Ports are used to send messages to the item managers in charge of control of tasks, devices, memory, and other resources and are used to enforce the principle of "least privilege" in TMach.

6.3.4 Use of Mach memory maps

Mach memory mapping is used to make available different kinds of resources to TMach tasks. As seen in 6.2.2.2, files are memory mapped to the task's virtual address space. Ports are similarly mapped. In fact, all items managed by item managers that the user needs to be able to directly manipulate can be mapped into the user's address space or port space. This provides a uniform method for obtaining access as well as assurance that unmapped objects cannot be accessed except through messages routed through ports to the item servers that manage the objects.

54.[23] page 9.

7. Allocation of TMach Mechanisms to Security Services

This chapter discusses the use of the TMach mechanisms, as described in Chapter 6, to implement the various DGSA concepts. None of the TMach servers were modified, and only one new server is added to the TCB, although several others must be trusted to properly enforce the security policy and to implement required services. In particular, the TMach Audit and Authentication Servers [23] are used as defined.

We first describe this implementation, including the management of information domains and of security contexts within those domains. We then look at how well this implementation works for the DGSA requirements of strict isolation, separation of policy definition from enforcement, security associations, and provision of security services. We conclude with an examination of the resulting TMach implementations of MLS policies and the guard example presented in Chapter 4.

7.1 TMach Implementation

This section describes an implementation of the DGSA using the primitive entities in TMach. These include tasks, ports, messages, and memories, and are managed by the kernel (which may itself be thought of as a task). Ports, which are accessed through rights stored in a task's port name space, are used as object names in addition to being a communication mechanism through which messages are passed. In particular, each task is named by a task port, rights for which the task and its parent are given when it is created. Likewise, the kernel is named by a host port, the rights for which are held by a small set of trusted tasks.

The set of information domains for a collection of secure applications is managed by a TMach task called the *Information Domain Server* (IDS). A collection of applications is started by creating an IDS. The IDS is responsible for creating and destroying information domains, and for transferring objects between them. A request for one of these operations is sent from an information domain through the *IDS request port*. The IDS start-up code creates the initial information domains.

Each information domain is managed by an *Information Domain Manager* (IDM) task. The IDS creates a new information domain by creating its IDM. It initializes the IDM port name space with a send right to the IDS request port; thus, the IDS is known by each of the IDMs and there is no reason to name it through the TMach Root Name Server (RNS). The IDM address space is initialized with code to process requests from users, enforce the security policy, and create a security context for each user of the domain. This code must be trusted, but it is largely shared among all IDMs and can be carefully tested (the names of the files providing any nonshared code, including the processing of specific requests, would be parameters). The IDS destroys an information domain by terminating its IDM task, using the send right for the task port that it got when it created the IDM.

Each information domain is known to potential users by a symbolic name. An IDM's initialization code enters this name in the RNS and associates it with a send right to its *descriptor port* that it creates to receive object descriptors from the RNS. The Access Control List (ACL) for the domain stored in the RNS identifies the operations permitted and the auditing required for each user in the domain; it thus forms part of the SMIB. A user enters a domain by first looking up the domain name through the RNS. If that user is listed in the ACL for the domain, an *object descriptor* for the user is sent to the domain IDM and a send right for the descriptor is returned to the user. The descriptor contains the user's id and the permissions that the ACL grants to the user. Upon receipt of the object descriptor, the IDM creates a *context task* to serve as the security context for that user and associates the object descriptor with this task in a context list. It also saves the task port for the context and send rights for a port through which it may send requests to the context. It maps the code that implements operations defined for the domain into the virtual memory of the new context task, and places receive rights for the context's request port in the initial context port name space.

When an IDM receives a request from the user, it compares the Security Id of the user issuing the request with the Security Id in the object descriptor and enforces the security policy specified as a function of the accesses for the user that are stored in the object descriptor. The IDM itself handles requests to create or destroy objects, or to modify the set of users of the domain. It sends requests to transfer an object, or to create or destroy an information domain, to the IDS request port. All other allowed requests are sent to the request port of the context task associated with the object descriptor from which the request was received.

An information object is implemented as a TMach memory object managed by the IDM of the domain to which the object belongs. An object can either exist in the virtual address space of the IDM, or else be made more permanent than the IDM by storing it in the file system.

A context task requests an object by sending the object's name to the IDM *context port*, which is created by the IDM initialization code and for which a send right is placed by the IDM in the initial port name space of each context task that it creates. (The name used in this request is local to the domain and does not have to correspond to a file name.) The requested object is mapped into the context's virtual address space.

An object is transferred between domains by the IDS. When a user requests a copy of an object from another domain, the IDM receiving the request (after verifying that the user has import permission) sends the name of the object, the name of the providing domain, and the Security Id of the user to the IDS request port. The IDS gets an object descriptor for the providing domain from the RNS and sends a request for the copy (containing the object name and the user) to the object descriptor. The providing IDM checks that it has the object and that the user has permission to copy it, then sends either the copy or a rejection to the IDS, which forwards the reply to the requesting domain. Note that if the object is maintained as a file, a "copy" can be just the file name which the receiving IDM then opens through the RNS and maps to the contexts that request it.

Likewise, a user can request that an information object be exported to another domain. The IDM receiving the request sends the name of the object, the name of the receiving domain, the Security Id of the user, and either a copy of the object or else the memory object representative port, to the IDS request port. The IDS forwards the user, the name of the object, and the object to an object descriptor in the receiving domain. This domain can either accept or reject the object. If the information object exported from a domain should be removed from that domain, the object must first be deallocated from each context virtual memory to which it has been mapped, also flushing it if it is backed by a file.

The user set for an information domain is maintained by the RNS in the ACL for the domain's IDM. The IDM implements requests to change this set by modifying its ACL. If a user is removed from the set, the IDM must also terminate any context tasks for the user, using the send right for the task port saved with the object descriptor for the user.

The communication paths between the various TMach tasks are shown in Figure 12. Some of the ports used are shown in dashed boxes. An IDM can receive:

- requests — from users in that domain (using object descriptors provided by the RNS),
- objects transferred to the domain — from the IDS (using a port created by the IDM when requesting the transfer),

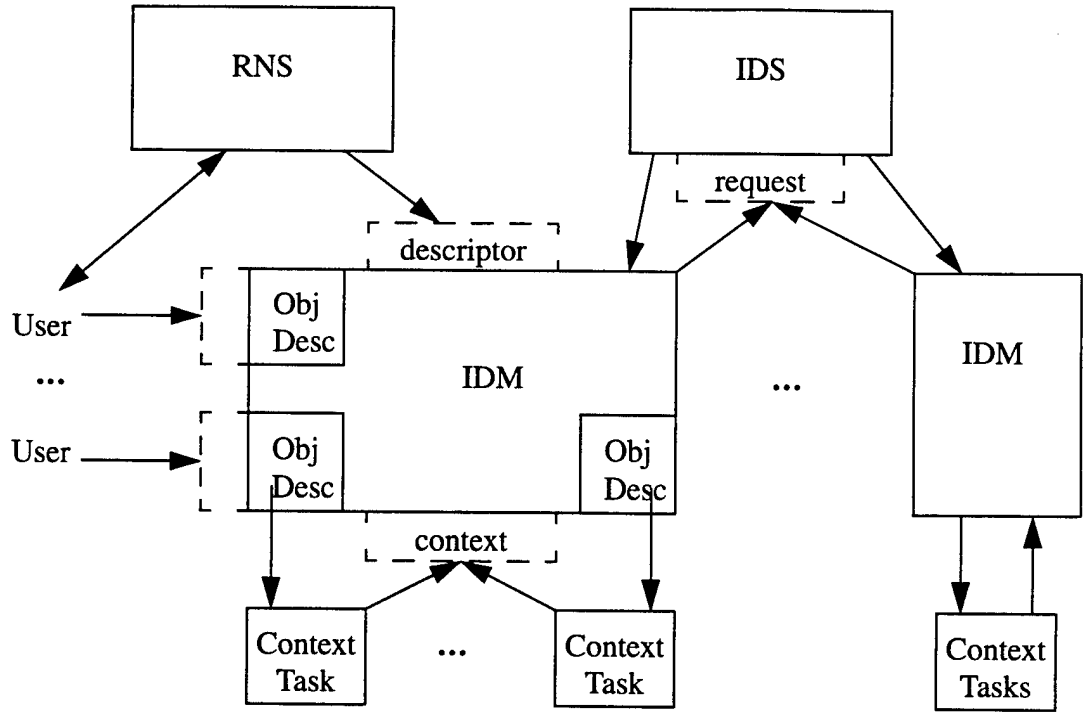

Figure 12. TMach Tasks and Communications Paths for the DGSA

- requests for objects and for domain operations — from the contexts (using the context port created when the IDM is initialized),
- object descriptors — from the RNS (using the descriptor port created when the IDM is initialized), and
- file system objects — from the RNS (using ports created by the IDM when requesting the object).

An IDM can send:

- domain management and object transfer requests — to the IDS (using the IDS request port in the initial IDM port name space),
- user requests — to the contexts (using ports that it creates for those contexts and places in their initial port name space),
- domain access information and file names — to the RNS (using the IDM bootstrap port), and

- file requests — to the file server (using ports received from the RNS or the IDS).

7.2 Strict Isolation

The security contexts must be strictly isolated from each other, except for the sharing of objects by contexts of the same information domain. This isolation can be provided without trusting the context tasks. TMach tasks can communicate either through shared memory or through ports. Initially, a context's port name space only contains receive rights for its request port through which its IDM issues requests (for which only the IDM has send rights). Also, the bootstrap port is set to the IDM's context port; it has no means for issuing requests to the RNS. Its virtual address space only has executable code. Thus, it can only communicate with the kernel and the IDM, and cannot map any memory objects into its address space (since it does not have rights for any memory object representative ports). We require that the IDM not send any port rights to a context (any port rights must be removed from user requests before they are sent to a context task) and that the IDM not send its rights to the context request ports to any other task. Also, the IDM may only map information objects of the domain that it manages into a context task virtual address space, and it must deallocate any mapped objects when removing them from the domain.

While the contexts can be untrusted, the IDM is a key to maintaining an isolated domain and must be trusted. It must properly manage any mapped areas of the virtual memory for the context tasks that it creates. It must not allow these tasks to receive any new ports, and must only send them allowed requests. Finally, it must only transmit objects to the IDS (in response to valid export commands) or to the appropriate file. File system security can be used to help separate domains by giving ownership of each file used to implement objects to the IDS and setting the ACL for the file so that only the IDM for the domain owning the supported object may access a file (each IDM would require a separate Security Id), but the IDM must still be trusted not to continue using a file after it has transferred the object to another domain.

7.3 Separation of Policy Decisions and Enforcement

The DGSA requires that mechanisms for security policy enforcement be separate from those for security policy decisions. In the proposed TMach implementation, both of these mechanisms are contained in the IDMs (decisions are supported by the ACLs contained in the RNS, but these are set by the IDMs). To meet the requirement, an IDM contains a Security Policy Decision Function (SPDF) that is called by the code that enforces the policy. The SPDF is a memory object that is mapped into the virtual memory of the IDM and can therefore be changed by changing the mapping. The parameters to the SPDF include the request and the

accesses allowed to the user making the request. The output includes the decision on whether the request should be allowed and whether it should be audited. This design has been shown to be very flexible and is capable of implementing a large set of policies [20].

An alternative design is to use a separate Security Server for the SPDF [20]. This server could serve several Information Domains and its inputs would include the security attributes for the domain requesting a decision. This design requires communication between the IDM and the Security Server on every request, which can significantly impact the performance of the system. This performance loss can be mitigated by caching policy decisions in the IDM, provided that there is a means for revoking decisions for use with history-based security policies.

7.4 Security Associations

A security association links Information Domains that have the same security policy, but which may be on different end systems. Each such association is managed by a Security Association Manager (SAM) that is created by the IDS in response to a *create association* command from an IDM. Each domain in the association is represented by an *association port* that its IDM presents to the SAM on joining the association. The SAM uses these ports to broadcast messages to the members. A Network Server on each end system maps ports for nonlocal IDMs across the network to the end system on which the IDM resides.

The members of a security association must reside on end systems that provide services sufficient for the protection required for the domain. In addition, the security policy of each member must be the same. Thus, each request to join a security association must be authenticated, possibly with a cryptographic key provided by the security administrator of the end system on which the IDM resides. Likewise, the SAM must authenticate itself to each IDM that requests membership.

When an IDM receives a request for an object about which it has no knowledge, it passes the request to the SAM which broadcasts the request to all of the IDMs in the association. Each IDM either responds negatively, or else provides a memory object representative port for the object that is then used to map the object into the requesting context's virtual memory. The IDM that provides the object acts as the memory manager and must implement a coherence algorithm to maintain consistency among multiple users of the object.

The set of users in the information domain must be maintained consistently across all of the associated IDMs. When an IDM joins the association, the SAM obtains the current set of users from the other members (some translation to the user Security Ids of the new system

may be necessary). This becomes the user set of the new IDM. Changes to the set of users of a domain must be broadcast to all of the IDMs in the association.

7.5 Basic Services

7.5.1 Access Control

In TMach, access control is split between a decision function defined by the RNS and an enforcement function defined by the servers. The RNS maintains an ACL for each information domain, set by the IDM, to define the accesses allowed by the security policy. When a user looks up a domain name, the RNS provides it with a descriptor based on the permissions in the ACL for that user, and provides the IDM with the Security Id of the user that will use that descriptor (so that the user cannot give its permissions away to another user). The IDM checks any requests that it receives against the permissions in the descriptor used to send the request before forwarding it to the context for processing. If a user is removed from the domain, the IDM terminates its contexts. Thus, the strength of access control depends on the correctness of the IDM in setting up the domain ACL, enforcing access restrictions when requests are made, and terminating contexts when users are removed from a domain.

7.5.2 Confidentiality

Information should only be visible as authorized by the security policy. Information within a domain is maintained in regions of memory within an IDM or as files that are only accessible by the IDM, and is mapped into the virtual address spaces of context tasks controlled by the IDM. Since the TMach memory management provides strong confidentiality, the strength of confidentiality within a domain is equal to that with which isolation is maintained, as described in Section 7.2. Within a domain, the transfer of information between objects is carried out by operations requested by users. Thus, the strength of confidentiality depends on the extent to which the code contained in the contexts properly implements the operations.

Information can only be exported by an IDM to the IDS (for transfer to another IDM or to a device) or to another IDM with which a security association has been established, as described in Section 7.4. Thus, confidentiality also depends on the strength of confidentiality provided by the IDS (a trusted task) and the mechanism used to communicate between associated IDMs. If both IDMs are on the same end system, the TMach communication mechanism is used, which provides strong confidentiality. Otherwise, confidentiality depends on the strength of confidentiality provided by the Network Servers and the Communication Network.

7.5.3 Integrity

Information should only be modified as authorized by the policy. The considerations described for confidentiality also apply to integrity. To allow for security policies in which certain users cannot modify information in a particular domain, but only read information and transfer objects, contexts must be established that cannot modify information objects, even if the operations are incorrectly implemented. This is done by the IDM by not providing the task port to the context when it is created (so that the context cannot modify the maximum protections in its virtual address space) and by mapping objects into the domain as read only.

7.5.4 Availability

Availability is only indirectly provided. TMach is meant to run on a single end system and is not fault tolerant. Protection against processor failures can be provided through the security association mechanism. A policy in which each object in a domain is held by two or more associated IDMs can be made part of the coherence algorithm. If an IDM and its contexts fail, the objects are still available.

The process scheduler, part of the kernel, determines when each task is allowed to execute. TMach is not a real-time system for which guaranteed service can be set. However, the scheduling algorithm and the task priorities are system parameters, and the IDS could be given enough privileges to set them.

7.5.5 Authentication

TMach authenticates individuals creating tasks through a device using passwords. The individual invokes a Trusted Shell using the Trusted Attention Key. The Trusted Shell then requests identification and a password, which it verifies using the Authentication Server. The Trusted Shell then creates a task with security attributes returned by the Authentication Server.

Tasks spawned from other tasks are given the same Security Id as the parent, unless the parent task is part of the TCB. Therefore, a task is identified with the individual who is ultimately responsible for its creation.

Security associations across a network depend on the associated IDS and IDMs authenticating each other. On a single end system, TMach authenticates the sender of a request to the receiver. When the communication extends over a network, the Network Servers must forward the identity of the sender with a message. The trust of this identification depends on the strength with which the Network Servers are able to authenticate each other.

7.6 Audit

As the security policy enforcers, the IDMs have primary responsibility for collecting audit information. Requests from users to perform operations are checked first by the IDM. It thus has knowledge of any requests that are rejected, of the users making requests, and of the permissions being used. A local Audit Condition List for the domain, indexed by user, permission, and request acceptance, defines the audit policy and is contained in the SPDF described in Section 7.3. Thus, the audit policy can specify all accesses by a particular user, or all rejected requests. It cannot specify all accesses to a particular object, however, since all objects in an information domain have the same security attributes and cannot be individually identified by the policy.

Processing audit information can be facilitated by the TMach Audit Server [23]. This server will either store events in the file system, or send them to an alert device. It can also filter audit events using a remote Audit Condition List.

7.7 TMach Implementation of MLS Policies

TMach does not allow tasks with different MLS levels to communicate unless one of them is part of the TCB [24]. Thus, to avoid adding potentially dynamically-created IDMs to the TCB, we require that an IDM and all of its users and contexts have the same MLS level, consistent with the maximum level of the information objects in the domain. Unless all of the IDMs have the same level, however, the IDS must be able to create tasks and transfer information objects of multiple levels and therefore must be part of the TCB.

An MLS security policy normally allows users to read from objects at a lower security level and write to objects at a higher level. In a DGSA-style system, reading an object from another domain is accomplished by exporting a copy of the object into the current domain. Similarly, writing to another domain is accomplished by exporting the object. These transfers are allowed by TMach since they are made by the IDS, which is part of the TCB and operates at multiple levels. While the security policies enforced by the IDMs should only allow the transfer if the level of the sending domain is dominated by the level of the receiving domain, the IDS will normally also enforce this restriction as a safety measure.

7.8 TMach Implementation of the Guard Example

This section describes how some of the guard domains defined in Chapter 4 would be implemented in TMach. Initially, the guard consists of the IDS task in the TCB. The IDS starts the guard by creating IDM tasks for the Source Connection, the Regraded Block, each Block

Type, and the PiL Record information domains. Each of these IDMs has the same MLS level as the Source user, except for the Regraded Block, which has the same level as the Destination user. The IDS may also start tasks to fill the roles of Dispatcher, Router for each block type, Filters, and Sanitizers, or these tasks may be started by the SSO. The IDMs register with the RNS, also establishing the access control policy in the ACL. Rather than establishing an Audit domain, the Audit Server is used.

The Dispatcher works in the Source Connection domain, entering it by getting an object descriptor from the RNS. The Source Connection IDM creates a Dispatcher context task that handles Dispatcher requests to process an information block (transferred from the Source user to the IDM by the IDS). The Dispatcher context task repeatedly:

- sends a request to the Source Connection IDM for an information block (which is mapped into its address space),
- determines the type of the block,
- sends a request to the IDM that a Block Evaluation domain of the determined type be created (this request is forwarded to the IDS), and
- sends a request to the IDM that the block be exported to the new domain (this request is forwarded to the IDS).

Since the only action by the Dispatcher user task is to send an initial request, it can be combined with the Source Connection IDM. Also, since there is only one context, this can be combined with the IDM task. The result is a Source Connection task that receives information blocks produced by the Source user by way of the IDS and that sends requests to the IDS.

A Block Evaluation IDM task is created by the IDS is response to a request from the Dispatcher task. This task will accept an information object transferred from the Dispatcher. It registers with the RNS, specifying an initial ACL that allows the Router user for the type of block to import from the Block Type domain and to process the block. The Router user:

- gets an object descriptor for the domain from the RNS, which also causes the IDM to create a Router context task,
- sends a request to the IDM for a copy of the filter/sanitizer list from the Block Type domain, causing a transfer request to be sent by the IDM to the IDS,
- sends a request to the IDM to process the block, which is forwarded to the Router context task, and

- sends a request to the IDM to terminate the domain, which is forwarded to the IDS.

The Router context task processes the block by first requesting the filter/sanitizer list (which the IDM maps into its address space) and then either sending a request to the IDM to transfer the block to the PiL (this request is forwarded to the IDS) or it:

- sends a request to the IDM for the signature list (which is mapped into its address space),
- steps through the filter/sanitizer list:
  - adds the next to the domain by sending a request to the IDM,
  - waits until the filter/sanitizer appends to the signature list,
  - removes the filter/sanitizer from the domain by sending a request to the IDM, and
  - looks for a rejection notice in the domain and, if there is one, sends the notice and the information block to the Audit Server and terminates.

Rather than create separate tasks for the filter and sanitizer users, these users are made part of the Block Evaluation IDM tasks. When an IDM receives a request from a Router context to add a filter or sanitizer, instead of adding that user to the ACL and waiting for the RNS to send an object descriptor, it just creates a new context to which it sends a request for processing.

For both of the domains described, users that only operate in that domain (except for transferring blocks) were combined with the IDM task, slightly complicating the IDM but removing the need to describe the sequencing and synchronization of operations in a user task. There are still many context tasks that must be created and managed, but these provide the address space separation required by the DGSA.

8. Findings and Recommendations

8.1 DGSA Support in Commercial Operating Systems

Trusted Information Systems' TMach has a number of admirable and useful properties. The question remains, however: How useful is it as a base for DGSA? This question is especially important, since we expect any limitations to be present in most other commercial systems. Table 2 summarizes some of the limitations and the corresponding remedies that fix them.

Table 2. TMach Functional Support To DGSA Based OS

	Degree or Qualification	To Fix/ Fixable
Platform Support	Limited	Must provide on other platforms, e.g., SPARC
OS Personalities	Unix Only	Requires NT 4.0 Personality
Devices	Very few	Requires overhaul of TMach to simplify the addition of new devices without re-evaluation
Networking	Single sensitivity level	Requires Security Association implementation
Security	Strict Isolation provided	Requires IDS & IDM and certification for Integrity, etc.
Distribution	None	Requires Distribution Server for domain objects
COTS/GOTS	Poor	Requires NT 4.0 Personality
Fault Tolerance Reliability & Availability	None	Requires overhaul of TMach in order to achieve
Operational Assurance	Minimum (Audit Server)	Need to determine requirements
Evaluation	Abandoned	Need B3 or B3/F5 on original TMach + IDS, IDM, SAs and Distribution Server + new drivers

As has been described in previous chapters, TMach provides many basic functions useful in constructing an implementation of DGSA-compliant applications. In particular, it provides the means for enforcing strict isolation, which is the key to implementing information domains. However, the design presented did not make natural use of the TMach facilities. Also, there are several problems that we did not adequately address. Some of the problems are due to TMach's design, some to its limited implementation, and some to the way in which we chose to use TMach features to implement DGSA-compliant applications. The following paragraphs outline the difficulties that we found.

8.1.1 Support for Platforms

TMach is only supported on an Intel 486/66DX2 platform with a very limited set of peripherals. To be truly useful in the DoD, TMach must accommodate the following additional platforms (the base set for the DII): the full range of Intel X86 platforms, IBM Corporation RS-6000, Digital Equipment Corporation Alpha, Hewlett Packard PA-RISC, and Silicon Graphics platforms; or be compatible with versions of these produced after 1995.

8.1.2 Support for Operating System Personalities

TMach only supports a POSIX personality that approximates a Unix interface. On the version running on the Intel platform, DoD requires a personality supporting the Microsoft Win32 interface, and in particular a Microsoft NT Version 4.0 personality must be provided so that current Windows 95 applications may be run *as is*.

8.1.3 Support for Devices

TMach does not easily support the insertion of new device drivers. This is in part due to the fact that device drivers must be embedded in the microkernel, which is part of the TCB. Each driver must be rigorously proven correct in order to maintain trust that the driver does not decrease the degree of isolation between domains, and that it labels its output appropriately. Since DoD would desire to support any device that is supported in Windows NT, support for devices is an especially difficult problem.

TMach requires that a security officer manually change keys used by its encryption device. This requirement is a burden if each domain has its own encryption key. Therefore, an API is required in support of IPSEC that provides programmed change to encryption keys.

8.1.4 Support for Networking

Support in TMach for networking exists at the transport level for single-level networks through untrusted servers. Networking security policies in the DGSA occur at the application level, as represented by the security associations for each distributed domain, and may require several domains to share one network. A multiplexor, trusted to enforce availability requirements, must be provided.

8.1.5 Support for Security

The client-server paradigm on which TMach is built proved appropriate for the development of DGSA. We had little difficulty in supporting the security policies for the guard outlined in Chapter 4. Table 3 provides a summary of support provided by TMach for DGSA.

Table 3. Support of TMach for DGSA Security

	Level/Kind	To Fix/Fixable
Strict Isolation of Domains	High	No fix required
Authentication	Weak (by password)	Replaceable Modules
Confidentiality	High (Local)	SA, IDM
Integrity	Low (by isolation)	IDM, SA, FT(?)
Non-repudiation	Low (by principal id)	IDM, SA
Availability	None	Need Requirements
Audit	Medium (Audit Server)	Need Requirements
Regrading	None	IDS
Multi-domain objects	None	Scripting Language
Security Associations	None	Development of SA
Separation of Policy & Enforcement	Strong (Confidentiality) Weak (others)	IDM, IDS, Distribution Manager, Authentication Implementation
I/O, e.g., printing, X-Windows	Medium (labeling)	?
History based policies	None	IDM, IDS, SA

TMach has substantial support for isolation, but none for integrity policies, non-repudiation policies, or policies that use histories to make decisions. The sensitivity brackets and

categories used for isolation in the current system are of marginal utility in the DGSA. They can be used for read-down operations that permit use of a single copy of a program on disk instead of one per domain. When the program is invoked, it will be read up and instantiated in the domain of the invoker and then run in that domain.

Separation of decision and enforcement is high for confidentiality, but is low for other aspects of security policy such as authentication, signing, non-repudiation, availability, etc. Authentication is weak as it relies on passwords. It is not "replaceable" as it stands. An IDM might have additional authorization components that can be used, but additional software is required to read biometrics or smartcards. Identifiers, such as those of objects and security contexts, are locally rather than globally unique.

8.1.6 Support for Distribution

TMach does not support distribution, e.g., no distributed information object management or security association is provided. (This is provided in a follow-on research vehicle called Triad.) In order to be satisfactory, a new distribution server would be required. If the system is only to be used in a homogeneous environment, this would be a moderate size task; in a heterogeneous environment, this could be a major task.

8.1.7 Support for COTS and GOTS

TMach has a personality emulation for POSIX, but does not have a personality emulation for WIN32 libraries or NT system functions.

8.1.8 Support for Fault Tolerance, Reliability, and Availability

No support for parallel processing, redundancy, fault tolerance, or real-time is provided in TMach. (Some of these features are provided in a followon research system known as Triad.) These aspects play a particularly important role in availability policies in distributed systems.

8.1.9 Support for Operational Assurance

TMach has no support for operational assurance except for an audit server. A suite of tools that attempt to exploit TMach's structure with a variety of threats would be very useful.

8.1.10 Required Supplements to TMach

An implementation of DGSA requires that the system support an API and a TCB Server for regrading (like the IDS) that implements inter-domain transfers. This server must be trusted

to enforce the transfer policies of the sending and the receiving domains, as expressed in their SMIBs.

If the security policy for a domain provides for more than absolute protection and inter-domain transfer, e.g., if it has an integrity component, the implementation requires a TCB server (like an IDM) acting as security context manager for that domain. Ideally, this manager is table driven from information in the domain's SMIB. The context management could be done by tasks of the IDM, one task per security context as described in Section 7.1. While the IDM require no special privileges, they must be certified to implement the policy aspects required of them and they must be authenticated as implementing the desired security policy.

If distributed operation is desired, the implementation requires that a sub-task of the IDM exist for each active domain. This assures that information objects that are distributed become accessible on any machine that wishes to operate on them, effectively virtualizing the information objects as far as the networked systems are concerned. Also the implementation requires that there be one or more TCB servers acting as manager of security associations forming the distributed security context. This server interacts closely with the individual distribution managers and requires access to security policy on intermachine transfers.

Additional work needs to be done in order to provide alternate methods of authentication and to enable separation of decision and enforcement in the event that a security policy specifies an area other than protection.

8.1.11 Evaluation

One of the reasons that TMach was chosen was that it was being evaluated in the U.S. at the B3 (COMPUSEC) level and in the European Union at the B3-E5 (INFOSEC) level. Subsequently, these evaluations have been abandoned. If the evaluations had certified TMach at the B3 level, one would have substantially more confidence in the fact that TMach provides the basis for strict isolation. If possible, any base operating system considered for implementation of DGSA should be evaluated favorably at this level[55], including all necessary components.

---

55. Unfortunately, Microsoft NT Version 3.5.1 was only evaluated at the C2 level and only when physically isolated from the network. Version 4 has not been evaluated to the best of our knowledge. The addition of Mandatory Access Controls (MAC) and additional design verification of a B3 system add to our confidence that strict isolation can be enforced.

8.2 DGSA Weaknesses

The DGSA must address the software architecture, in addition to the hardware architecture and the data architecture, if implementors are to be guided in their implementation. Table 4 addresses areas of DGSA that could be improved in such a way as to aid implementors

Table 4. To Aid DGSA Implementors

	Status/Degree of Definition	To Fix/Fixable
Policy	Undefined	Need policy classes covering the policy space, differing in parameterization
— Types of Policies to be Employed	Unbounded	Need bounding for assurance
— Policy Representation	Undefined	Want table/rule driven
— Placement of Policy Information	Vague	Need guidance
— Dynamic Policies	Undefined	Needs development
— Threats	Vague	Needs development
— Support of DGSA for MLS	Definition of DGSA includes aspects of MLS	Develop efficient isolation mechanism preventing covert channels
Model of Computation	Undefined	Needs development
Security Association	Vague	Requires immediate clarification
Information Object	Vague	Needs development
Information Domain	Well Defined	Nothing
Absolute Protection	Vague	Needs quantification
Evaluation	Undefined	Needs development
Multi-domain objects	Vague	Needs development
Security Labels	Vague	Standardization (FIPS?)
Sharing Information Objects	Vague	Determination of shareability

without unnecessarily foreclosing implementation options. Version 3 of the DGSA is very terse and abstract when it discusses the logical entities such as information object and "behaviors" that affect them. It is terse and extremely concrete when it discusses physical entities such as end systems. Thus, the DGSA is a partial data architecture and a partial systems architecture. The implementor is faced with a design space that is barely constrained and that does not discuss current topics of portability or interoperability except for transportation of information objects via Security Association. Although this level of constraint appears to have been deliberately chosen by the architects, the current document leaves too much to the imagination of implementors.

8.2.1 Policies

A better specification of the classes of policies to be enforced (especially for integrity and non-repudiation) is required. Whether a given implementation will be capable of handling every possible security policy (that could differ in Identification and Authentication (I&A), Access Control, Confidentiality, Integrity, Non-Repudiation, Availability, and Assurance characteristics) cannot be determined.

Determining whether a set of security policy profiles that provide generic, parameterized policies is desirable. The implementation of the services and mechanisms for enforcing the policies can then be done once and reused with the minimum expenditure of effort during accreditation and certification. Advance, generic certification should be provided if table values are within acceptable ranges for a new domain using this policy structure. This will greatly reduce the administrative load that will normally accompany the instantiation and administration of a domain.

A policy satisfying MLS requirements (and that can be implemented efficiently in information domains) is needed. The DGSA and the current MLS policy may not be well suited to one another. In an MLS system, a user may operate simultaneously with information of several sensitivity levels and categories, and with different access rules for each. In the DGSA, a user works only with information objects that share a set of security attribute values, and the accesses allowed are the same for each object. To simulate an MLS policy, the objects that may be observed must first be copied into a domain representing an upper bound on the sensitivities and compartments. Any objects created or modified can then be transferred back to the original domain. Conversely, operating system support for the DGSA needs several features of the MLS model, but not all. Especially important are strict isolation and suppression of any covert channels. However, the hierarchical relationship between the levels is of little use for the DGSA.

8.2.2 Computational Model

The DGSA provides no constraint on the computational model except strict isolation. In particular, it says nothing about the interoperability of the computational model over distributed platforms except that information objects may be shared between end systems. An example computational model covering both local and distributed computations should be added to the defining document or to a guidance document, not as a specification, but to show that at least one computational model is sufficient to do the task.

The DGSA endorses absolute protection, but fails to provide any guidance about how to measure it. This is a critical concept if multiple hardware platforms using multiple instantiation of DGSA are to be joined by security associations. A more quantitative notion of absolute protection should be developed and added to the report or the guidance document.

8.2.3 Evaluation and Operational Assurance

The DGSA does not provide any details about how an implementation is to be evaluated. Such a discussion would be very helpful to the designers because the evaluation criteria will direct the design to obtain the highest score in evaluation without expending design or implementation resources needlessly.

The DGSA does not provide any details about the class of threat models that will be employed against its features. A parameterized threat model would be useful to the implementor in focusing attention on what measures must be taken to counter the threat. A common threat model, such as proposed by Rothkopf and Mallett, would be very valuable.

The DGSA provides few if any details about requirements for operational assurance, a key concept in a system that handles domains of arbitrary sensitivity. A discussion should be added to the document, or referenced, that describes how to determine the amount of operational assurance needed, and how to determine whether the amount available is sufficient.

8.3 General Recommendations

The DGSA seems to be a valuable framework for designing a practical security architecture that can be used in both commercial and military environments. The following sections deal with additional work, or information that might be added to or referred to by the DGSA definition, description, guidance, and reference.

8.3.1 On Further Prototyping of DGSA

A working implementation of DGSA is needed as soon as possible in order to convince the Services and Agencies that the architecture is feasible and has utility. In order to accomplish this objective, several important subsidiary objectives must be achieved.

- Profiles of security policies likely to be used are needed; a profile of services drawn from ISO 10181 should be provided that satisfies the policies. Research is needed to determine whether sufficient policies can be accommodated with a table-driven or rule-driven set of managers.
- A standard for security labeling is required that will be used to assure interoperability of labels when a system or systems must distinguish between information at different levels, or when separated from the information domain or security association with which the information is associated. The current FIPS on security labeling would appear to be an appropriate starting point for implementation.
- An application to application security association is urgently needed. The Internet Protocol Security (IPSEC) specification draft gives an IP-level implementation, with the possibility of extension to the transport level. This would be sufficient if only one domain were hosted on an end system. In order to provide the isolation required, an application to application security association must be developed that can utilize information about the security policy. Security policy profiles relative to the security associations must be defined and, if possible, relegated to a table driven form.
- Work is required to develop a scripting mechanism compatible with DGSA. The scripting mechanism would permit "invocations" to be submitted to a scripting execution mechanism within each domain on behalf of users or delegates for users. Condition codes or opaque descriptors to information objects could be returned and could be used in addition to strings of characters as parameters to domain methods. The "system entity" could retain named state. This mechanism would be used to represent multi-dimensional objects and sequence and to co-ordinate their printing, transmission, sequencing, exportation, etc., without inappropriate direct access to the information.
- Work is needed on an evaluation mechanism for absolute protection. Without the evaluation mechanism, showing that two implementations on different platforms

have enough strength to permit the implementation to be accredited will be impossible.

- A reference implementation that works with applications from the Microsoft Windows compatible group of applications is required. If applications from this suite cannot be transparently executed in an information domain, the DGSA likely will be ignored.
- An official conformance test for a DGSA-compliant system is needed. This test must accommodate as wide a class of systems as possible that conform to the DGSA architecture. An officially approved procedure that leads to the certification of an information domain on a system is required. This will be important when the domain must be certified on multiple end systems so that users can work together using shared information objects.
- The subset of MLS requirements for the platforms used to provide DGSA must be determined.

8.3.2 Increasing the Utility of the DGSA Definition

Without the possibility of a reference implementation, the DGSA report is not a useful document, nor is it likely to be taken seriously. The definition needs an example done at the implementation level. The example does not need to be a full implementation, but does need to be complete enough so that skilled implementors will concede the feasibility of an implementation. This example should be referenced in the report.

Discussion of Policy, Risk, Threats, and Assurance as they relate to the architecture should be more fully covered in the document or a companion document. Without this additional information, security professionals are less likely to give credence to the DGSA.

8.3.3 Difficult Areas of DGSA

More examples need to be provided in DGSA documentation, or in guidance documents referenced by the DGSA report, that show:

- How security associations work, including a sketch of all the major subfunctions that are required, e.g., key management, global authentication, encryption and its relationship to security policy, etc.
- How global policy can be translated into domain policies and how those domain policies can be composed to form the system policy.

- How distributed information object management might work within the framework of the DGSA (as opposed to within CORBA, etc.).
- How security policies other than those based on confidentiality, such as those based on state, integrity, non-repudiation, availability, etc., might be specified and/or implemented.
- How dynamic security policies can be integrated into the DGSA.

A body of DGSA experts should be identified who can provide qualified interpretation of the document to (potential) implementors (similar to the group that interpreted the early Ada specification and conformance tests). Even with the examples referenced above, the document as it stands can lead implementors to add unneeded functionality or to produce implementations that will not satisfy the DGSA when evaluated. The experts can help to assure that only required functionality occurs in implementations of the DGSA.

A curriculum consisting of educational materials on the DGSA would be a valuable addition for those intent on implementing systems that conform to the DGSA. Too few people are adequately informed about its potential applicability for DoD applications and the implications of the DGSA on system architecture.

8.4 The Future of DGSA

If the DGSA is to move beyond a specification into concrete implementations, the following specific steps should be taken:

1. Develop a transition plan for incorporating the DGSA into system implementations. Without such a plan, the DGSA will not be used.
2. Develop a plan to determine MLS features required by the DGSA implementation and to see if they can be implemented on a wide variety of computing platforms, e.g., Virtual Machines with minimal capacity covert channels and Security Associations such as IPSEC.
3. Establish a body that can issue authoritative interpretations of DGSA. Without such a body, there will be few implementations.
4. Develop a plan in coordination with the commercial sector to turn DoD Goal Security Architecture into a National Goal Security Architecture for use in the commercial sector as well as the defense sector. Without such a plan, commercial vendors will not implement DGSA.

5. Implement a prototype DGSA as soon as possible that presents a Microsoft NT API (perhaps limited) on a single machine: this is required for broad acceptance in the commercial community. Demonstration of a commercially viable DGSA base which supports COTS applications is required before DoD and the Services will be willing to accept DGSA.

6. Establish a class hierarchy for security policies and the parameter ranges required for the first DoD and Commercial deployment. This is imperative in order to narrow the number of implementations which must be produced for the IDMs. One such class might be those policies required for electronic commerce.

7. Determine if a database-driven security policy decision maker is feasible and if so how the policies may be allowed to change dynamically. This would dramatically lower the cost of DGSA deployment and operation, likely spurring the production of DGSA compliant systems.

8. Provide incentives for the design and implementation of a suitable version of security associations for the second level of deployment. This could result in issuance of IETF specifications leading to widespread commercial implementation.

9. Explore how well the IETF IPSEC security association meets the needs expressed here. It may be possible that only a small amount of additional funding is required to obtain a military grade solution generated by the commercial sector.

10. Determine metrics for end system protection. These will be required in order to evaluate implementations and to accredit domains.

11. Determine the procedure for information domain certification on one machine and on a distributed system of machines. This will determine what constitutes absolute protection.

12. Develop a methodology for demonstrating that a dynamic set of components satisfies a set of security requirements. This is required to assure that adding an existing domain to an additional LSE or adding a new domain to an existing LSE does not require a new assurance test for accreditation.

13. Provide more detailed guidance on implementation without foreclosing varieties of implementation needlessly. This is necessary to avoid cost overruns or poor performance in implementation caused by too loose a specification of what is desired.

Acronyms

ACL	Access Control List
API	Application Programing Interface
ASSR	Agreed Set of Security Rules
CIIF	Common Imagery Interoperability Framework
CIIWG	Common Imagery Interoperability Working Group
CISS	Center for Information System Security
CN	Communication Network
COMPUSEC	Computer Security
DISA	Defense Information Systems Agency
DGSA	DoD Goal Security Architecture
DISSP	DoD Information Systems Security Program
DoD	Department of Defense
DOS	Disk Operating System
EDS	Entity-Domain-System
ES	End System
IBM	International Business Machines
IDA	Institute for Defense Analyses
IETF	Internet Engineering Task Force
INFOSEC	Information Security
IP	Internet Protocol
IPMO	INFOSEC Program Management Office
IPSEC	IETF's Internet Protocol Security Protocol
ISAKMP	Internet Security Association and Key Management Protocol
ISMC	Imagery Standards Management Committee
ISO	International Standards Organization
LSE	Local Subscriber Environment
NSA	National Security Agency
PiM	Person in the Middle

POSIX	
RS	Relay System
SA	Security Association
SAK	Secure Attention Key
SAMP	Security Association Management Protocol
SMAP	Security Management Application Processes
SMIB	Security Management Information Base
SPDF	Security Policy Decision Function
SPEF	Security Policy Enforcement Function
SSO	System Security Officer
TAFIM	Technical Architecture Framework for Information Management
TCB	Trusted Computing Base
TIS	Trusted Information Systems
TMach	Trusted Information Systems' Trusted MACH®
USIGS	United States Imagery and Geospatial System

References

[1] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. on Programming Languages and Systems 15*(4): 706–734, September 1993.

[2] W. E. Boebert and R. Y. Kain. A practical alternative to hierarchical integrity policies. In *Proceedings 8th National Computer Security Conference*, pages 18–27, Gaithersburg, MD, October 1985.

[3] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, April 1987.

[4] National Security Agency. *Department of Defense (DoD) Information Systems Security Policy*, DISSP-SP.1. February 1993.

[5] Defense Information Systems Agency, Center for Standards. *Department of Defense (DoD) Goal Security Architecture (DGSA),* Version 3.0, April 1996.[56] Volume 6 of [6].

[6] Defense Information Systems Agency, Center for Standards. *Department of Defense Technical Architecture Framework for Information Management (TAFIM),* Version 3.0, April 1996.

---

56. The DGSA is available through http://www-library.itsi.disa.mil/tafim/tafim3.0/pages/tafim_6.htm and may be downloaded in a variety of forms. Paper and floppy disk distribution is available through the Defense Technical Information Center (DTIC) — call 1-800-225-DTIC for credit card or NTIS account orders, or 703-274-6871 to get on their user registry. The DGSA is available on floppies in Word Perfect 5.1/5.2 - Order No. AD-M000 303, in PostScript - Order No. AD-M000 304, or Macintosh Word - Order No. AD-M000 305. DTIC has recently established an anonymous FTP Internet server asc.dtic.dla.mil (131.84.1.22). The DGSA is also available on Dockmaster in WordPerfect or ascii (no figures, not recommended) in >site>net>papers>dgsa. The DGSA is available from the NIST Security BBS in WordPerfect and PostScript formats. The NIST BBS is accessible by anonymous FTP Internet service at csrc.nist.gov (129.6.54.11) and by dial-up modem (9600 baud) at 301-948-5140. For FTP, the files are located in the /bbs/secpubs directory. For download the files are located in #13 directory. Refer to V6README.TXT for a list of the individual files.

[7] David Garlan and Mary Shaw. An introduction to software architecture. In *Advances in Software Engineering and Knowledge Engineering*, edited by V. Ambriola and G. Tortora, World Scientific Publishing Company, 1993.

[8] Morrie Gasser. *Building a Secure Computer System.* Van Nostrand Rheinhold Company, New York, ISBN 0-442-23022-2, 1988.

[9] *Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.* International Organization for Standardization, ISO/IEC 7498-2, 1989.

[10] *Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management Framework.* International Organization for Standardization, ISO/IEC 7498-4, 1989.

[11] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 1: Overview.* International Organization for Standardization, ISO/IEC 10181-1, 1996.

[12] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 2: Authentication.* International Organization for Standardization, ISO/IEC 10181-2, 1996.

[13] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 3: Access Control.* International Organization for Standardization, ISO/IEC 10181-3, 1996.

[14] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 4: Non-Repudiation.* International Organization for Standardization, ISO/IEC DIS 10181-4, 1996.

[15] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 5: Confidentiality Framework.* International Organization for Standardization, ISO/IEC 10181-5, 1996.

[16] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 6: Integrity Framework.* International Organization for Standardization, ISO/IEC 10181-6, 1996.

[17] *Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems — Part 7: Security Audit and Alarms Framework.* International Organization for Standardization, ISO/IEC 10181-7, 1996.

[18] Bill Nitzberg and Virginia Lo. Distributed shared memory: A survey of issues and algorithms. *IEEE Computer 24*(8): 52–59, August 1991.

[19] *MK++ Kernel Executive Summary*. Open Software Foundation, November 1995.

[20] Duane Olawsky, Todd Fine, Edward Schneider, and Ray Spencer. Developing and using a "policy neutral" access control policy. In *Proceedings of the New Security Paradigms Workshop*, Lake Arrowhead, CA, September 1996.

[21] John Rushby. A Trusted Computing Base for embedded systems. In *Proceedings of the 7th DOD/NBS Computer Security Symposium*, pages 294–311, September 1984.

[22] E. John Sebes et al. The architecture of Triad: A distributed, real-time, trusted system. In *Proceedings 17th National Computer Security Conference*, pages 237–246, Baltimore, MD, October 1994.

[23] *Trusted Mach System Architecture*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0001-97A. March 1997.

[24] *Trusted Mach Philosophy of Protection*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0003-96A. October 1996.

[25] *Trusted Mach System Administration Guide*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0014-97A. March 1997.

[26] *Trusted Mach Security Features User's Guide*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0015-97A. March 1997.

[27] *Trusted Mach Protection Mechanisms*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0032-95A. September 1995.

[28] *Trusted Mach Trusted Path Command Reference Guide*. Trusted Information Systems, Glenwood, MD, Document No: TIS TMACH Edoc-0033-96F. November 1996.

[29] Common Imagery Interoperability Working Group(CIIWG) of the Imagery Standards Management Committee (ISMC). *CIIF Security Services Analysis*. National Photographic Interpretation Center, National Exploitation Laboratory. September 1996.

[30] —. *Integrity in Automated Information Systems*. National Computer Security Center, Fort George G. Meade, MD. September 1991.

APPENDIX A. Definitions, Axioms, and Constraints

Definitions

Definition 2.1: A *security policy* $p \in \mathbf{P}$, where **P** represents the universal set of all security policies, is a collection of rules designed to enforce an organization's desired protection for its information and other system resources, and to counter given threats.

Definition 2.2: An *information object* $o \in \mathbf{O}$, where **O** represents the universal set of all information objects, is a unit of information that is atomic with respect to security policies. The type of an information object defines the effect of the various accesses allowed by the security policy.

Definition 2.3: A *user* $u \in \mathbf{U}$, where **U** represents the universal set of all users, is a unit of responsibility for accesses to information objects and information system resources. Each user is represented by an information object containing its credentials.

Definition 2.4: An *information system resource** $r \in \mathbf{R_t}$, where $\mathbf{R_t}$ represents the universal set of all information system resources at time t, is a provider of information system services and/or facilities for processing, transmission, and storage (e.g., input/output devices, memory, registers, system functions).

Definition 2.5: A *security attribute* $a \in \mathbf{A}$, where **A** represents the universal set of all security attributes, is an element of information that is associated with an entity for the purpose of applying a security policy. Each security attribute has an associated set of values.

Definition 2.6: Let A_e be the set of possible values of the security attributes for entity e and

$$E_{OBJECT} = \{ (o, a) \mid o \in \mathbf{O} \wedge a \in A_o \},$$

$$E_{USER} = \{ (u, a) \mid u \in \mathbf{U} \wedge a \in A_u \},$$

$$E_{RESOURCE}(t) = \{ (r, a) \mid r \in \mathbf{R_t} \wedge a \in A_r \}.$$

Then a *controlled entity** $e \in (E_{OBJECT} \cup E_{USER} \cup E_{RESOURCE}(t))$ is an information object, user, or resource that has associated security attributes.

Definition 2.7: An *information domain security policy* $p \in \mathbf{P_D}$, where $\mathbf{P_D} \subset \mathbf{P}$ is the universal set of information domain security policies, is a statement of the criteria for membership of users in an information domain and the required protection, including conditions of use, for the information objects in the domain.

Definition 2.8: An *information domain* $d \in \mathbf{D}$, where **D** represents the universal set of all information domains, consists of a set $O \subseteq \mathbf{O}$ of uniquely identified information objects, a set $U \subseteq \mathbf{U}$ of users, and a domain security policy $p \in P_D$. At time t it is denoted $d_t = \{O, U, p\}$.

Definition 2.9: An *information domain import-export policy* is that part of an information domain security policy that establishes the rules, conditions, and procedures for the transfer of information objects with other domains.

Definition 3.1: An *end system* $es \in \mathbf{ES}$, where **ES** represents the universal set of all end systems, is an information processing system to include processor and input/output devices (e.g., workstation, personal computer, server, minicomputer, mainframe, disk drive, printer, telephone) directly accessible by users.

Definition 3.2: A *local communications system* $lcs \in \mathbf{LCS}$, where **LCS** represents the universal set of all local communications systems, is a set of communication devices (e.g., ring, bus, twisted pair, coaxial cable, fiber-optic cable) under the direct (physical) control of a local subscriber environment.

Definition 3.3: A *relay system* $rs \in \mathbf{RS}$, where **RS** represents the universal set of all relay systems, is an information processing system (e.g., multiplexor, router, switch, cellular node, message transfer agent) not directly accessible by users that manages transfers of information between a public network and a local communications system.

Definition 3.4: A *local subscriber environment* $lse \in \mathbf{LSE}$, where **LSE** represents the universal set of all local subscriber environments, is a set $lse = \{ESL, RLS, LCSL\}$, where $ESL \subseteq \mathbf{ES}$ is an ordered list of end systems, $RSL \subseteq \mathbf{RS}$ is an ordered list of relay systems, and $LCSL \subseteq \mathbf{LCS}$ is an ordered list of local communications systems.

Definition 3.5: A *controlled entity collection** $ce_{(x,z)}(t) \in \mathbf{CE}$, where **CE** is the universal set of all controlled entity collections, is a set of controlled entities at time t obtained by holding the information domain on the x-axis and the end system on the z-axis constant, and taking the vertical projection along the y-axis.

Definition 3.6: An *information domain view* of information management at time t is defined by a set of controlled entity collections CE obtained by holding information domain d (on the x-axis) constant (i.e., selecting a particular information domain), and taking the vertical projection along the y-z plane. It is the union of controlled entity collections across information domain d.

$$CE_d = \bigcup_{1 \le j \le q} ce_{(d,es_j)}(t)$$

Conversely, a complementary view can be defined by fixing the end system.

Definition 3.7: An *end system view* of information management at time t is defined by a set of controlled entity collections CE obtained by holding end system es (on the z-axis) constant (i.e., selecting a particular end system), and taking the vertical projection along the x-y plane It is the union of controlled entity collections across end system es.

$$CE_{es} = \bigcup_{1 \le i \le v} ce_{(d_i,es)}(t)$$

Definition 3.8: *Strict isolation* is the absolute separation of a set of information domains, $D = \{d_i,...d_n\}$, n > 1, *and* their associated controlled entities from other domains on a specific end system at any instant of time.

Definition 3.9: Let $d \in$ **D** be an information domain and $es \in$ **ES** be an end system. A *security context* $sc_{(u,d,es)}(t) \subseteq ce_{(d,es)}(t) = O \cup U \cup R \cup \{\pi\}$ supporting $u \in U$ (where R is a set of resources and π is the implementation of p on es) is a set of controlled entities $O' \cup \{u\} \cup R' \cup \{\pi\}$, $O' \subseteq O$ and $R' \subseteq R$, forming an operating environment for u in domain d on end system es at time t.

Definition 3.10: A *security management information domain* $d_M = \langle O_M, U_M, p_M \rangle$ comprises a set of uniquely identified information objects $O_M \subseteq$ **O** grouped into security management information bases (SMIBs), a set of privileged users $U_M \subseteq$ **U**, and a management information domain security policy $p_M \in$ **P**.

Definition 3.11: An *end system security policy* π_{SYS} is a security policy that specifies requirements for sharing of information system resources (e.g., security functions, services, mechanisms, devices, memory, registers) on an end system es in support of a set of information domains resident on es.

Definition 3.12: For a given information domain d projected onto a set of end systems $ES = \{es_1,...,es_q\}$, q > 1, *absolute protection* is the condition of achieving the minimum required strength of protection for d on each $es \in ES$.

Definition 3.13: Let $d \in$ **D** be an information domain, $ES = \{es_1,...,es_q\} \subseteq$ **ES** ($q > 1$) be a set of end systems, and $CE = \{ce_{(d,es1)}(t),..., ce_{(d,esq)}(t)\}$ be a set of controlled entity collections. A *distributed security context* $dsc_{(u,d,ES)}(t) \subseteq \cup CE = O \cup U \cup R \cup \{\pi\}$ supporting $u \in U$ is a set of controlled entities $O' \cup \{u\} \cup R' \cup \{\pi\}$, $O' \subseteq O$ and $R' \subseteq R$, forming an operating environment for u in domain d on the end systems in ES at time t.

Definition 3.14: A *security association* is the set of all information system resources (i.e., security and communications protocols, security functions, security services, and mechanisms) employed to securely link two distinct security contexts, sc_i and sc_j, on different end systems, es_i and es_j, supporting the same information domain, d.

Axioms

Axiom 1: An information object cannot reside in two different information domains simultaneously.

Axiom 2: A user can be a member of multiple information domains simultaneously.

Axiom 3: An end system, relay system, or local communications system cannot reside in two different Local Subscriber Environments simultaneously.

Axiom 4: An instance of an information object may only exist on one end system at a time.

Axiom 5: Users may operate on more than one end system simultaneously.

Axiom 6: An information system resource can support (be a member of) more than one controlled entity stack, but only in a time-multiplexed manner.

Constraints

Constraint 1: Information objects must have the same set of policy-specified security attributes and the same security attribute values.

Constraint 2: Users within an information domain must have the same set of policy-specified security attributes but may have different security attribute values.

Constraint 3: The user responsible for an inter-domain transfer of information objects must be a member of both the source and the destination domains and must be permitted to make the transfer by the import-export policies of each domain.

Constraint 4: Inter-domain transfers of information objects can occur only if the source and destination information domains are resident on the same end system and if the transfer agent is a user in both domains.

Constraint 5: Distributed security contexts can support information transfer only within a single information domain.

Constraint 6: Each information domain providing a constituent information object for a composite multi-domain object must be supported on the end systems participating in the transfer of that multi-domain object.

APPENDIX B. ISO Bibiliography

International Organization for Standardization (ISO). 1989. *Information Processing Systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture.* ISO/IEC 7498-2: 1989.

International Organization for Standardization (ISO). 1989. *Information Processing Systems -- Open Systems Interconnection -- Basic Reference Model -- Part 4: Management Framework.* ISO/IEC 7498-4: 1989.

International Organization for Standardization (ISO). 1997. Information Technology - *Vocabulary - Part 8: Security* (Revision of ISO 2382-8:1986). ISO/IEC DIS 2382-8.

International Organization for Standardization (ISO). 1992. Information Technology - *Open Systems Interconnection - Systems Management: Security Alarm Reporting Function.* ISO/IEC 10164-7:1992.

International Organization for Standardization (ISO). 1993. Information Technology - *Open Systems Interconnection - Systems Management: Security Audit Trail Function.* ISO/IEC 10164-8:1993.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 1: Overview.* ISO/IEC 10181-1: 1996.

International Organization for Standardization (ISO). 1996, Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 2: Authentication.*ISO/IEC 10181-2: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 3: Access Control.* ISO/IEC 10181-3: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 4: Non-Repudiation.* ISO/IEC DIS 10181-4.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 5: Confidentiality Framework.* ISO/IEC 10181-5: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 6: Integrity Framework.* ISO/IEC 10181-6: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Security Frameworks in Open Systems - Part 7: Security Audit and Alarms Framework.* ISO/IEC 10181-7: 1996.

International Organization for Standardization (ISO). 1995. Information Technology - *Telecommunications and Information Exchange Between Systems- Transport Layer Security Model.* ISO/IEC 10736: 1995.

International Organization for Standardization (ISO). 1995. Information Technology - *Open Systems Interconnection - Upper Layers Security Model.* ISO/IEC 10745: 1995.

International Organization for Standardization (ISO). 1997. Information Technology - Open Distributed Processing - Reference Model: Foundations. ISO/IEC DIS 10746-2.

International Organization for Standardization (ISO). 1995. Information Technology - Open Distributed Processing - Reference Model: Architecture. ISO/IEC DIS 10746-3.

International Organization for Standardization (ISO). 1995. Information Technology - *Open Systems Interconnection - Network Layer Security Protocol.* ISO/IEC 11577: 1995.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Overview, Models and Notation.* ISO/IEC 11586-1: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Security Exchange Service Element(SESE) Service Definition.* ISO/IEC 11586-2: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Security Exchange Service Element(SESE) Protocol Definition.* ISO/IEC 11586-3: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Protecting Transfer Syntax Specification.* ISO/IEC 11586-4: 1996.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Security Exchange Service Element(SESE) Protocol Implementation Conformance Statement(PICS) Proforma.* ISO/IEC DIS 11586-5.

International Organization for Standardization (ISO). 1996. Information Technology - *Open Systems Interconnection - Generic Upper Layers Security: Protecting Transfer Syntax Implementation Conformance Statement(PICS) Proforma.* ISO/IEC DIS 11586-6.

International Organization for Standardization (ISO). 1996. Information Technology - *Guidelines for the Management of IT Security - Part 1: Concepts and Models for IT Security.* ISO/IEC TR 13335-1: 1996.

International Organization for Standardization (ISO). 1997. Information Technology - *Guidelines for the Management of IT Security - Part 2: Planning and managing IT Security (Future Technical Report).* ISO/IEC DTR 13335-2.

International Organization for Standardization (ISO). 1997. Information Technology - *Guidelines for the Management of IT Security - Part 3: Techniques for the Management of IT Security.* ISO/IEC DTR 13335-3.

International Organization for Standardization (ISO). 1995. Information Technology - *Open Systems Interconnection - Lower Layers Security Model.* ISO/IEC 13594: 1995.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1997	Final

4. TITLE AND SUBTITLE
Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms

5. FUNDING NUMBERS
DASW01-94-C-0054
Task Order T-AA5-1409

6. AUTHOR(S)
Edward A. Schneider, Edward A. Feustel, Ronald S. Ross

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Paper P-3375

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Security Agency
9800 Savage Road
Fort George G. Meade, MD 20755-6000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; unlimited distribution: 20 June 1998.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

Acceptance of the DoD Goal Security Architecture (DGSA) has been hindered by the fact that no implementation of a system embodying all aspects of the DGSA has been delivered, and many believe that such an implementation is impossible on currently available computing systems. We address those concerns in this report of our investigation of how the DGSA might be implemented using a commercially available operating system, Trusted Information Systems' Trusted Mach (TMach). This report first develops formal definitions for concepts defined in the DGSA and provides a model to describe implementation in the distributed case. It then extends the concepts in DGSA to describe features needed in the computational process. Next, a DGSA-style security architecture is created by describing a mission statement and developing a security policy and a set of components that support the mission. It then discusses the management of security services. Next, it discusses TMach and describes how TMach mechanisms are allocated to security services required by the security policy. Finally, it discusses areas of DGSA that TMach does not support, areas of DGSA that need refinement for purpose of implementation, and recommendations on the further development of the DGSA.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Defense Goal Security Architecture; Trusted Information Systems' Trusted Mach (TMach); Security Architecture; Security Policy; Information Domain; Distributed Computing; Security Association.	134
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL